UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

OBJETOS DE APRENDIZAGEM MULTIMÍDIA PROVENIENTES DA CAPTURA UBÍQUA DE APRESENTAÇÕES MULTIMODAIS: PRODUÇÃO, INTERAÇÃO E ANÁLISE

CAIO CÉSAR VIEL

ORIENTADOR: PROF. DR. CESAR AUGUSTO CAMILLO TEIXEIRA

São Carlos – SP
Abril/2013

UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

OBJETOS DE APRENDIZAGEM MULTIMÍDIA PROVENIENTES DA CAPTURA UBÍQUA DE APRESENTAÇÕES MULTIMODAIS: PRODUÇÃO, INTERAÇÃO E ANÁLISE

CAIO CÉSAR VIEL

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação, área de concentração: Sistemas Distribuídos e Redes
Orientador: Prof. Dr. Cesar Augusto Camillo Teixeira

São Carlos – SP

Abril/2013

Ficha catalográfica elaborada pelo DePT da
Biblioteca Comunitária da UFSCar

V659oa

Viel, Caio César.
Objetos de aprendizagem multimídia provenientes da captura ubíqua de apresentações multimodais : produção, interação e análise / Caio César Viel. -- São Carlos : UFSCar, 2013.
138 f.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2013.

1. Multimídia interativa. 2. Computação ubíqua. 3. Ensino a distância. 4. e-Learning. I. Título.

CDD: 006.6 (20ª)

Universidade Federal de São Carlos
Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Ciência da Computação

"Objetos de Aprendizagem Multimídia Provenientes da Captura Ubíqua de Apresentações Multimodais: Produção, Interação e Análise"

Caio César Viel

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Membros da Banca:

Prof. Dr. Cesar Augusto Camillo Teixeira
(Orientador - DC/UFSCar)

Profa. Dra. Vânia Paula de Almeida Neris
(DC/UFSCar)

Prof. Dr. Wilson Vicente Ruggiero
(EPUSP)

São Carlos
Maio/2013

À minha madrinha.

AGRADECIMENTOS

Gostaria de agradecer a todos que me ajudaram nesse longo caminho. Primeiramente aos meus pais, sem os quais nem estaria em São Carlos. Ao meu melhor amigo e companheiro de aventura, Rafael, por ter me apoiado o tempo todo e por tudo o que você fez e significa pra mim. Aos meus colegas de laboratórios Erick, Kamila, Tiago, Arthur, Alessander, Danilo, Daniel, Endril e Alexandre, cujas valiosas ajudas e momentos de descontração tornaram este trabalho realidade. Ao meu orientador e minha co-orientadora extraoficial, Cesar e Graça, por toda paciência, colaboração e incentivo. Aos meus grupos de RPG por me ajudarem a manter a sanidade. À parceiria com a UAB-UFSCar no decorrer do Projeto PRESENTE e ao CNPq e CAPES pelo apoio financeiro. Por fim, gostaria de agradecer a todos os professores, alunos e funcionários que colaboraram no projeto, seja ajudando a montar a sala instrumentada, seja participando de experimentos, ou com conselhos e sugestões.

“Sangue e juventude!” — urrou o jovem bandoleiro, e um trovão de vozes juvenis se ergueu em resposta. “Sangue e juventude!” — eram vozes malformadas, algumas ainda finas de infância. Eram bandoleiros ainda malformados, muitos dos corpos e todos os espíritos ainda crus. Mas eram desesperados.

O Crânio e o Corvo — Leonel Caldela

RESUMO

A evolução da computação ubíqua, das técnicas de manipulação de mídia contínua e da composição, representação e apresentação de objetos multimídia, entre outras áreas das tecnologias da informação e comunicação, viabilizam a exploração e o estudo de resultados interessantes que podem ser alcançados com a incorporação dessas tecnologias em diferentes áreas de conhecimento e de aplicação. Escolheu-se as áreas de *Electronic Learning* (EE) e *Massive Online Open Courses* (MOOC) para se realizar esse tipo de estudo e exploração. Além de propiciar um rico ambiente para pesquisas em educação, o trabalho requereu a elaboração de pesquisa aplicada em computação ubíqua, interação humano-computador e sistemas multimídia. O trabalho também envolveu o desenvolvimento e aperfeiçoamento de protótipos, particularmente de máquinas de apresentação e sistemas para captura & acesso de atividades humanas. Como subproduto, é apresentada uma sala de aula instrumentada capaz de realizar a captura ubíqua de apresentações educacionais multimodais para posteriormente transformá-las em objetos de aprendizagem multimídia. Esses objetos de aprendizagem podem ser acessados pela Web, via *Learning Management System* (LMS), tanto de forma assíncrona como em sessões síncronas entre professores/tutores/alunos. Os objetos de aprendizagem multimídia podem ser utilizados em qualquer contexto educacional (como MOOCs e no apóio ao ensino presencial). Além disso, pode-se utilizar o log das interações dos usuários com os objetos para encontrar possíveis alvos de melhoria nas apresentações educacionais.

Palavras-chave: Computação Ubíqua, EaD, MOOC, Educação Eletrônica, Captura e Acesso, Análise da Interação, Máquina de Apresentação, Sincronização de Mídia, Linguagens Declarativas

ABSTRACT

The evolution of ubiquitous computing, techniques for handling continuous media and composition, representation and presentation of multimedia objects, among other areas of information technology and communication, enable the exploration and study of interesting results that can be achieved with the incorporation of these technologies in different areas of knowledge and application. The areas of Electronic Learning (EE) and Massive Online Open Courses (MOOC) were chosen to perform this kind of study and exploration. Besides providing a rich environment for research in education, the work required applied research in ubiquitous computing, human-computer interaction and multimedia systems. The work also included the development and improvement of prototypes, particularly presentation machines and capture & access systems for human activities. As a byproduct, it presents an intrumentalized classroom which can perform the ubiquitous capture of multimodal educational presentation for further generation of an equivalent multimedia learning object. These learning objects can be accessed via Learning Management System (LMS) on the Web in an asynchronous way or in synchronous sessions with professors/tutors/students. The multimedia learning objects may be used in MOOCs and as support for classroom learning. Moreover, the log of users' interactions with the objects can provide clues for improvement in the educational presentation.

Keywords: Ubiquitous Computing, Distance Education, MOOC, E-Learning, Capture & Access, Interaction Analytics, Presentation Machine, Media Synchronization, Declarative Languages

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

Capítulo 1

INTRODUÇÃO

1.1 Contexto

As Tecnologias da Informação e Comunicação (TIC) vem sendo utilizadas nas últimas décadas no apoio a atividades de ensino. Primeiramente apenas Tecnologias da Informação permitiram o uso do computador em atividades offline, facilitando o acesso a conteúdos e jogos educacionais previamente produzidos e gravados em mídias como CDROM, e a implementação de programas de ensino em que a capacidade de processamento pode ser usada para orientação no progresso dos estudos, entre outras finalidades. O avanço nas tecnologias de comunicação, principalmente com as facilidades que vieram a ser oferecidas pela Internet e pelos sistemas Web, permitiu que novos paradigmas pudessem ser desenvolvidos e aplicados no apoio ao ensino. O modelo de educação a distância (EaD) fundamentado em comunicação assíncrona entre alunos, tutores e professores, ganhou grande impulso e é hoje o modelo predominante de EaD, além de ser também importante no apoio ao ensino presencial.

Além disso, os *Massive Open Online Course* (MOOCs), que são cursos oferecidos pela Internet a decenas ou centenas de alunos, vem se popularizando nos últimos anos. Importantes universidades, como a *Stanford University* [1], *Massachusetts Institute of Technology* [2] e *Havard University* [3], estão disponibilizando seus cursos pela Web. Entretanto, apesar da flexibilidade oferecida por MOOCs e EaD, a diminuição da interação presencial professor-aluno desses modelos educacionais, quando comparada com a da Educação Presencial, pode ser fator importante em possível redução do aproveitamento dos estudos e de aumento na evasão de alunos.

Com uma utilização mais intensa das TICs disponíveis, pode ser possível melhorar essa

[1] Stanford Engineering Everywhere – http://see.stanford.edu/see/faq.aspx

[2] MIT OPEN COURSEWARE - http://ocw.mit.edu/index.htm

[3] Havard Open Courses – http://www.extension.harvard.edu/open-learning-initiative

interação e, sobretudo, contornar as impossibilidades inerentes da ausência do contato presencial, com o oferecimento de recursos e facilidades para o aluno de MOOCs, não encontrados na Educação Presencial. Além disso, quando a comunicação e interações são mediadas por computador, como acontece normalmente na EaD ou MOOCs, são abertas oportunidades ímpares para coleta, armazenamento e processamento de informações que podem resultar em benefícios para alunos e professores, contando inclusive com características de personalização.

Materiais didáticos de qualidade ou com interação entre aluno e professor mais intensa, como videoconferências, ou mesmo aulas[4] gravadas e posteriormente reproduzidas pelos alunos, são recursos valiosos no apoio a MOOC. Entretanto, a produção de vídeo-aulas ou a realização de videoconferências apresentam alguns problemas e desafios como elevados custos operacionais ou restrições na largura de banda.

1.2 Motivações e Objetivos

Foi observado que cursos de EaD apresentam altos índices evasão (ABED, 2009). Dentre as possíveis razões que podem motivar a evasão dos alunos, destacam-se algumas encontradas na literatura [(SANTOS et al., 2008), (OLIVEIRA; CAVALCANTE; GONCALVES, 2012), (SACERDOTE, 2010)]:

- Falta de comunicação face-a-face entre alunos e professores;
- Ausência de material didático impresso (necessidade de ler no computador);
- Dificuldades com leitura e escrita por parte dos alunos; e
- Carência de material em vídeo de qualidade.

Materiais em vídeo produzido para cursos de EaD ou MOOCs normalmente são monótonos e/ou compostos de apenas um fluxo de áudio/vídeo. Isso torna diferente a experiência de assistir uma vídeo-aula de participar de uma aula presencial, em que o aluno pode interagir com o professor ou escolher onde deseja focar sua atenção (slides, lousa, livro-texto, etc.).

Mesmo para alunos presenciais, as experiências do candidato e de seu orientador mostram que alunos reclamam de longas aulas expositivas. Além disso, aulas presenciais não respeitam o ritmo de aprendizagem dos alunos. Alunos com dificuldades em entender o conteúdo podem

[4] No contexto deste trabalho, o termo aula ou *lecture* refere-se a um objeto educacional consistindo principalmente de apresentação, por parte do professor ou de um especialista, semelhante a uma aula expositiva presencial.

não conseguir acompanhar uma explicação, enquanto alunos com maior facilidade ou algum conhecimento prévio sobre o assunto podem ficar entediados caso o professor se estenda demais em algum tópico (BONWELL, 1996).

Apresentações multimídia interativas, como aquelas que podem ser construídas utilizando-se linguagens como *Nested Context Language* (NCL) (ABNT, 2007) ou *Synchronized Multimedia Integration Language* (SMIL) (BULTERMAN; RUTLEDGE, 2008) ou em plataformas como a Adobe Flash [5] e Microsoft Silverlight [6], podem ser compostas por múltiplos fluxos de áudio e vídeo, além de imagens e texto. Tais tecnologias permitem especificar sincronizações temporais e espaciais entre as mídias, além de permitir sofisticadas formas de interação. Devido as suas características, apresentações multimídia podem ser utilizadas para a construção de ricos objetos de aprendizagem.

Motivado pelos problemas apontados, tanto em MOOCs quanto no ensino presencial, bem como pelas oportunidades oferecidas por apresentações multimídia interativas, pretende-se, com os resultados desta pesquisa, tornar a produção de material educacional em vídeo simples para os professores. Para isso, propõe-se uma abordagem que se apoia no paradigma, bastante conhecido dos professores, de aula presencial expositiva.

Um dos problemas a ser contornado é a gerencia da razão custo/qualidade na produção do conteúdo multimídia. A necessidade de técnicos, de operadores de câmeras, de diretor de vídeo para as seleções de tomadas, da edição posterior do material, e de outros recursos comuns à produção de programas de TV podem encarecer muito ou mesmo tornar inviável o produto que se deseja realizar. Regra geral acaba-se por optar por uma configuração reduzida de pessoal, penalizando a qualidade do produto final e possivelmente sua eficiência para os propósitos educacionais.

Mesmo investindo-se o suficiente para garantir uma produção de qualidade, outros dois problemas devem ser apontados. Primeiro, a possível e provável insuficiência de capacidade de comunicação das redes de dados até os alunos, para que diversos fluxos de vídeo de alta qualidade possam ser apresentados ao vivo durante uma videoconferência. Este problema também poderia ser resolvido com investimento financeiro (eventualmente também de alunos, se forem participar de suas casas). Entretanto, mesmo garantindo-se qualidade na produção e na transmissão ou reprodução, o vídeo e seu áudio associado são capazes de registrar apenas parte da experiência de uma aula. Ainda que múltiplas câmeras sejam utilizadas na captura de imagens, apenas um fluxo de vídeo é gerado, refletindo o ponto de vista e decisões do diretor da produ-

[5] Adobe Flash Plataform – http://www.adobe.com/devnet/flashplatform.html

[6] Microsoft Silverlight – http://www.microsoft.com/silverlight/

ção. Em outras palavras, isso significa que, diferente do que acontece em uma aula presencial, o aluno vai ter que assistir que o diretor, e não ele ou o professor, achar importante no momento. Em uma sala de aula presencial, o aluno é quem decide onde focar sua atenção, se no professor, no slide, no quadro branco ou outra apresentação que estiver sendo realizada.

A geração de apenas um fluxo de vídeo, a utilização de poucos operadores de câmeras (geralmente apenas um) e a dispensa de diretor de vídeo e de edição posterior induzem o professor a limitar sua apresentação ao uso de poucos recursos em videoconferências ou gravações em vídeo de uma aula, o que pode empobrecer o objeto de aprendizagem e deixá-lo pouco eficaz.

Como objetivos específicos, este trabalho busca:

- Instrumentalizar uma sala de aula de forma a realizar capturas de apresentações educacionais de forma pervasiva, com equipamentos como câmeras, computadores, microfones, etc.
- Produzir objetos de aprendizagem multimídia a partir de mídias capturadas de forma ubíqua de apresentações educacionais;
- Disponibilizar aos alunos de MOOC objetos de aprendizagem que os façam ter uma sensação mais próxima ou semelhante a estar "presentes" em uma sala de aula;
- Incorporar facilidades aos objetos de aprendizagem multimídia, como navegação por contexto, anotação síncronas com o tempo, etc. (algo impossível em meios não mediados por computador), a fim de enriquecer ainda mais a interação com os objetos educacionais.
- Permitir aos professores e tutores apresentarem aos alunos, de forma síncrona, os objetos multimídia produzido, a fim de aumentar o contato face-a-face;
- Explorar linguagens declarativas para sincronização de mídia na produção de apresentações multimídia altamente interativas; e
- Analisar a forma como os alunos interagem com os objetos de aprendizagem multimídia a fim de encontrar indícios para os professores de como melhorar suas apresentações educacionais.

1.3 Metodologia de Desenvolvimento do Trabalho

*Foi realizada uma pesquisa experimental, com o desenvolvimento de protótipos de ferramentas para produção, interação e apresentação de objetos de aprendizagem multimídia.*Utilizando

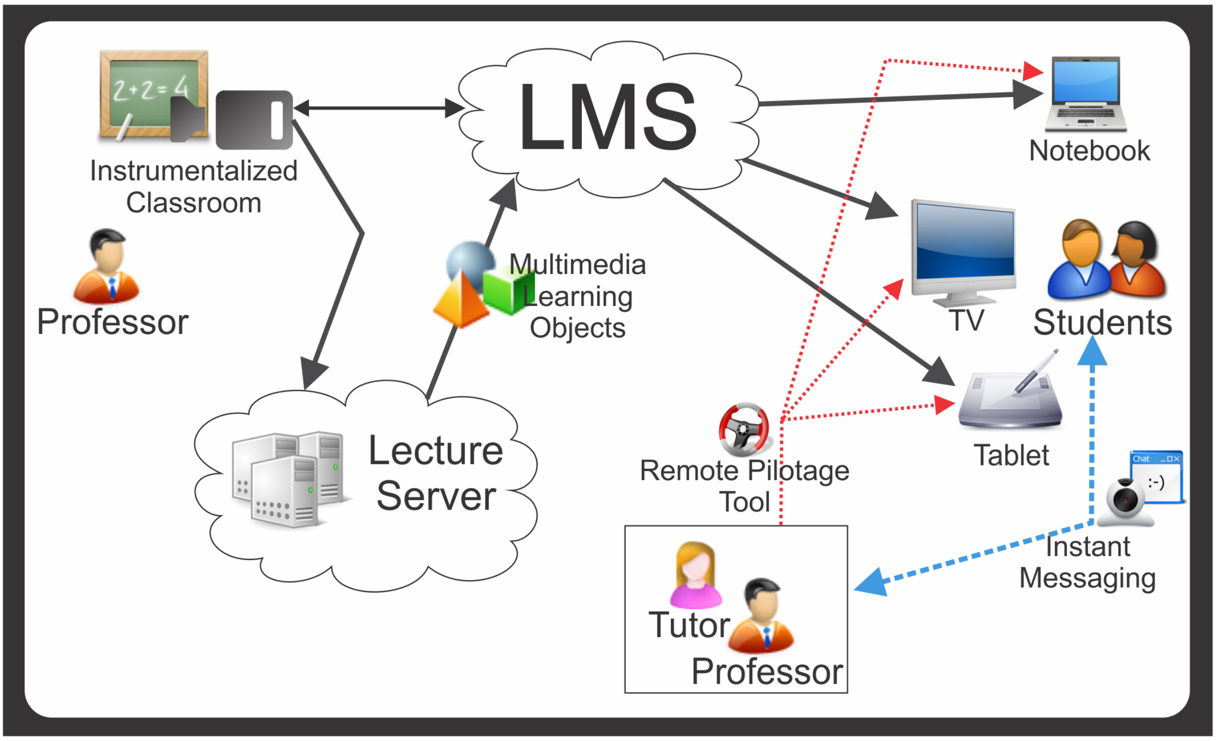


Figura 1.1: Visão geral do Projeto PRESENTE

essas ferramentas, bem como os objetos de aprendizagem multimídia gerados, foram realizados estudos de caso com professores e alunos dos cursos do Departamento de Computação da Universidade Federal de São Carlos.

Utilizando-se os dados coletados nos estudos de casos, bem como o *feedback* dos professores e alunos envolvidos, as ferramentas foram aprimoradas e novos estudos de caso foram realizados. Esse processo de desenvolvimento-testes-melhoria foi repetido ao longo de todo o trabalho. O aperfeiçoamento das ferramentas facilitou a realização dos estudos de caso. Através dos estudos de caso buscou-se atingir os objetivos propostos por esta pesquisa.

Este trabalho foi motivado por demandas reais decorrentes do Projeto BSI-PRESENTE – Produção e Apresentação de Objetos Educacionais Orientadas pelo Paradigma da Aula Presencial e pela Computação Ubíqua.

1.4 Projeto PRESENTE

A Figura 1.1 apresenta uma visão geral das ferramentas desenvolvidas no escopo do Projeto PRESENTE.

Na sala de aula instrumentada (*Intrumentalized Classroom*) apresentações educacionais realizadas pelos professores são capturadas e posteriormente analisadas no *Lecture Server*, onde o objeto de aprendizagem multimídia é gerado (*Multimedia Learning Object*).

Optou-se por adotar a linguagem NCL para representar os objetos de aprendizagem, pois NCL é uma das linguagens utilizadas para a produção de aplicações para o *International Integrated Services Digital Broadcasting* (ISDB-T), o sistema de TV Digital (TVD) utilizado no Brasil e em outro países da América Latina. Objetos de aprendizagem descritos em NCL podem ser exibidos em Set-Top-Boxes (STB) compatíveis com o ISDB-T. Para e exibição dos objetos em computadores pessoais convencionais, foi desenvolvido a WebNCL, uma máquina de apresentação NCL que executa em navegadores compatíveis com HTML5.

Os objetos de aprendizagem podem ser incorporados a *Learning Management Systems* (LMS) como o Moodle [7], TIDIA-Ae [8] ou Blakcboard [9]. No contexto do projeto PRESENTE, foi implementada a integração com o Moodle, que é o LMS adotado pelos cursos de EaD da UFSCar.

Os objetos educacionais, disponibilizados pelo LMS, podem então ser acessados através de diferentes dispositivos, como computadores e dispositivos móveis com navegadores HTML5 ou STB.

Além do modelo de interação assíncrono, em que o aluno pode interagir com o objeto de aprendizagem como e quando quiser, existe a possibilidade de professores e/ou monitores realizarem uma apresentação síncrona dos objetos através das ferramentas de Pilotagem Remota (*Remote Pilotage Tool*) e comunicador instantâneo (*Instant Messaging*), com suporte a chat por texto e videoconferência.

1.5 Organização do Trabalho

Este trabalho está organizado como uma coletânea de artigos escritos pelo candidato, mostrando contribuições de seu trabalho em diferentes áreas da computação. Os capítulos 2, 3 e 6 são artigos publicados em conferências. O capítulo 4 é um artigo aprovado em conferências, mas ainda não publicado. Não se tinha até o momento da publicação desta dissertação o resultado de aceitação do artigo, submetido a uma conferência, referente ao capítulo 5. No capítulo 7 a dissertação é concluída com a síntise das contribuições e trabalhos futuros.

As informações bibliográficas referentes aos artigos que compõem esta dissertação, bem

[7] Moodle – https://moodle.org/

[8] Tidia-Ae Aprendizado eletrônico – http://www.lipacs.iar.unicamp.br/projeto.php/1/tidia-ae-aprendizado-eletronico/

[9] Blackboard Learn Plataform – http://www.blackboard. com/Platforms/Learn/Products/Blackboard-Learn.aspx

como suas relações no contexto do trabalho, são apresentadas a seguir:

Capítulo 2: Viel, C.C.; Melo, E.L.; Pimentel, M.G.; Teixeira, C.A.C. **Presentations Preserved as Interactive Multi-video Objects**. In: *Proc. Workshop on Analytics on Video-Based Learning*, 2013. (VIEL et al., 2013c)

O Capítulo 2 apresenta em detalhes o modelo para captura de apresentações educacionais e geração de objetos de aprendizagem multimídia. O protótipo que implementa esse modelo também é descrito, bem como alguns estudos de caso realizados com professores e alunos.

Capítulo 3: Melo, E.L.; Viel, C.C.; Teixeira, C.A.C.; Rondon, A.C.; Silva, D.P; Rodrigues, D.G.; Silva, E.C.. **WebNCL: a Web-Based Presentation Machine for Multimedia Documents**. In: *Proc. Brazilian symposium on Multimedia and the web*: ACM, 2012. (WebMedia '12), p. 403-410. ISBN 978-1-4503-1706-1. http://doi.acm.org/10.1145/2382636.2382719 (MELO et al., 2012)

O Capítulo 3 apresenta a WebNCL, uma máquina de apresentação de documentos NCL desenvolvida utilizando a pilha de tecnologias Web (HTML5/JavaScript/CSS3) que possibilita a apresentação de documentos NCL em navegadores que suportam HTML5. A WebNCL é utilizada para permitir que alunos possam interagir com os objetos de aprendizagem multimídia em computadores pessoais convencionais. Esse artigo foi premiado como melhor artigo do *18th Brazilian Symposium on Multimedia and the Web* [10].

Capítulo 4: Viel, C.C.; Melo, E.L.; Pimentel, M.G.; Teixeira, C.A.C. **Go beyond boundaries of iTV applications**. In: *Proceedings of the 2013 ACM Symposium on Document Engineering*, 2013. (DocEng '13). (VIEL et al., 2013a)

Algumas das funcionalidades que foram sugeridas pelos alunos durante os estudos de caso realizados não eram triviais de serem implementadas em NCL. O Capítulo 4 discute essas funcionalidades e as estratégias utilizadas para implementá-las. Foram desensolvidos componentes para auxiliar na produção dos objetos de aprendizagem multimídia, esses componentes são apresentados e a utilização deles é ilustrada através da geração dos objetos de aprendizagem.

Capítulo 5: Viel, C.C.; Santos, T.P.C.; Queiroz, A.F.C.; Melo, E.L.; Pimentel, M.G.; Teixeira, C.A.C. **An Approach for Controlling Synchronous Remote Instances of a Multimedia Presentation**. In: *Submitted to a conference*, 2013. (VIEL et al., 2013)

O Capítulo 5 discute a problemática de controlar, remotamente, múltiplas instâncias de apresentações multimídia interativas. Uma prova de conceito, baseada na WebNCL, é apresentada. É através dos resultados apresentados por esse capítulo que se faz possível a apresentação

[10] WebMedia2012 - Best Paper Award: http://sws2012.ime.usp.br/webmedia/best-paper-award.php

síncrona de objetos de aprendizagem multimídia. O capítulo inclusive utiliza o cenário em que um professor realiza uma apresentação síncrona a seus alunos como estudo de caso.

Capítulo 6: Viel, C.C.; Melo, E.L.; Pimentel, M.G.; Teixeira, C.A.C. **How are they watching me: learning from student interactions with multimedia objects captured from classroom presentations**. In: *Proc. International Conference on Enterprise Information Systems*, 2013. (ICEIS '13). (VIEL et al., 2013b)

O Capítulo 6 relata um estudo de caso realizado com estudantes da UFSCar interagindo com os objetos de aprendizagem multimídia. Lições aprendidas ao se analisar o log das interações são discutidas. Esse capítulo ilustra os tipos de informações que são possíveis ao se analisar o log das interações dos usuários, de forma a permitir a realização de pesquisas na área de IHC. Esse artigo foi premiado como melhor artigo na área de IHC do *15th International Conference on Enterprise Information Systems* [11].

1.5.1 Roteiro de Leitura

A organização desta dissertação como uma coletânea de artigos ajuda a visualizar suas contribuições em diferentes áreas da computação. Caso deseje lê-la dessa forma, seguindo a order dos artigos apresendados, recomenda-se que algumas seções sejam puladas para evitar redundância entre os artigos. As seções redudantes são: 2.5.1, 6.2, 6.3 e 6.4.

Caso deseje ler a dissertação em um formato mais tradicional, pode-se seguir o roteiro de leitura sumarizado na Tabela 1.1.

[11] ICEIS - Best Paper Awards: http://www.iceis.org/PreviousAwards.aspx

	Assunto Abordado	**Seções**
Levantamento Bibliográfico	Captura e Acesso de Apresentações Educacionais	2.2
	Máquinas de Apresentação	3.2
	Linguagens Declarativas para Multimídia	4.2
	Sincronização de Apresentações Multimídia Distribuídas	5.2
Modelos	Captura e Acesso	2.1, 2.3
	Pilotagem Remota	5.1, 5.3, 5.4
Protótipos	Ferramenta de Captura e Geração	2.4
	Ferramenta de Apresentação	3.1, 3.3, 3.4, 3.5
	Objetos de Aprendizagem Multimídia	4.1, 4.3, 4.4, 4.5
	Pilotagem Remota	5.5
Discussões	Estudo de Caso: Professores	2.5, 2.6
	Estudo de Caso: Alunos	6.1, 6.5, 6.6, 6.7
	Máquinas de Apresentação	3.6, 3.7
	Pilotagem Remota	5.6

Tabela 1.1: Roteiro de Leitura

Capítulo 2

Educational Presentations Preserved as Interactive Multi-Video Objects

The performance of a teacher in the explanation of a subject can be a rich experience that deserves to be preserved, which should consider the experience as a multimodal and multi-device exposition performed with the support of voice, gestures, handwriting, whiteboards, interactive whiteboards, slide projection, and web browsers. In this paper we detail the design rationale applied in the development of a prototype environment for the ubiquitous capture of live presentations and their transformation into an equivalent interactive multi-video learning object. The record of users interactions with the multimedia object, as illustrated by a case study, provide useful information which educators can count for several educational purposes. To compensate the losses of some sensations and interactivity, impossible to keep in the recorded version, important context information are captured. These information, incorporated into the multimedia learning object, enable the reconstitution of the presentation and its exploration in dimensions not achievable in the classroom, as semantic browsing, collaborative annotation, multiple visualization, replays, etc. This proposal meets the resource constraints that are common in educational settings, trying to automate the process as much as possible.

2.1 Introduction

The term *educational presentation* is related to lectures. However, in the context of this work the term educational presentation has a broader meaning that includes, in addition to traditional lectures, other activities such as problem solving, short duration educational objects to be applied into constructivist activities, among others. But, without loss of generality, the contributions of this paper are discussed focusing on the preservation of lectures. The major motivation for this work was the desire to explore the possibility of lecture preservation to enable the future playback of the live class experience. Although a playback is not the as the live experience, it can be complemented by facilities which are possible only in a computer aided environment. These are lectures that can be enjoyed by students in distance education courses, or can serve to enrich the traditional face-to-face classroom model. Moreover, the aim is to support the preservation not only of lectures that are captured while presented to students, but also of lectures that are specifically produced to be captured.

We are aware that there are strong divergences among educators as to the efficiency of the lecture format as a method of instruction in middle schools and higher education. As stressed by Bauerlein (2011), "one of the axioms of progressivism education is that the lecture format is an inefficient and, potentially, alienating method of instruction". Ross (2011), for example, states that "when I was younger, I used to say that it took 40 years for any change in significant higher education to take effect, because that was the time by when all the existing teachers would have retired. I now realize that I was not a cynic, but an optimist, since lectures are just the prevalent as they ever were". However, as Ross himself acknowledges, lectures are still widely used in all levels of education. As Schwerdt e Wuppermann (2011) state in their studies, "contrary to contemporary pedagogical thinking, we find students score higher on that standardized tests in the subject in which their teachers spent more time on lecture-style presentations than in the subject in which the teacher devoted more time to problem-solving activities".

Bonwell (1996) and Cashin (1985) are among the authors who points out advantages and disadvantages of the lecture model. Advantages include:

- Effective lecturers can communicate the intrinsic interest of the subject through their enthusiasm.
- Lectures can make use of materials not otherwise available to students.
- Lectures can be organized to meet specific needs of a particular audience.
- Lectures can present large amounts of information.
- Lectures can be presented to large audiences.
- Lecturers can model how professionals work through questions or disciplinary problems.
- Lectures allow the instructor maximum control of the learning experience.
- Lectures present little risk for students.

- Lectures appeal to those who learn by listening.

The disadvantages of the lecture include:

- Lectures fail to provide instructors with feedback on the extent of the student learning.
- Students are often passive in lectures as there is no mechanism to ensure that they are intellectually engaged with the content
- Students' attention wanes quickly after fifteen to twenty-five minutes.
- Information tends to be quickly forgotten when students are passive.
- Lectures presume that all students learn at the same pace and are at the same level of understanding.
- Lectures are not suited for teaching higher orders of thinking such as application, analysis, synthesis, or evaluation, for teaching motor skills, or for influencing attitudes or values.
- Lectures are not well suited for teaching complex, abstract material.
- Lectures require effective speakers.
- Lectures emphasize learning by listening, which is a disadvantage for students who have other learning styles.

Although the model of an interactive multi-video lecture as the one described in this work has the potential to overcome some of the disadvantages, such discussion is a subject for researches in the area of education and is not dealt with in this paper. Instead, the goal of the work reported here is to present the rational applied in the development of a prototype environment for the ubiquitous capture of live presentations, and their transformation into an equivalent interactive multi-video learning object.

In this work, capturing a presentation in its entirety means recording the audio and one or more video streams of the speaker, the images presented on the screen or projector, the writings and drawings made on whiteboards, and also capturing contextual information that is relevant to its reproduction. Although the majority of this information is transformed into synchronized video streams, the resulting multimedia learning object may also include some static media. Even so, it is still referred to as an interactive multi-video object, since multiple synchronized videos are its main components. Therefore, in this paper we call an interactive multi-video object the composition of several videos, audio and some static media, properly synchronized and with facilities for flexible interaction and browsing.

The classroom activity is the primary learning context in many courses (ABOWD et al., 1999), so capturing such activities, lectures in special, may be interesting for several reasons. From the attendee's perspective, a student may use the recordings to do homework or to study for an exam, or a student who misses a class may still have access to what was presented by watching the recordings. From the instructor's perspective, a professor who will be absent from the campus may prepare a recorded lecture to deliver to the students. Moreover, a previously captured lecture may be improved and reused, or a captured lecture, or short takes, may be used specifically as a complementary learning object in different educational approaches. Last but not least, captured lectures can be a valuable resource for e-learning and distance education courses (LIU; KENDER, 2004).

Although recording lectures is a common practice in universities, producing quality video lectures demands a high operational cost (cameraman, video director, editors and other audio-visual professionals). To reduce the operational cost, many tools for automatic capture lectures were developed in the past decades ((BROTHERTON; ABOWD, 2004), (NAGAI, 2009), (CHOU et al., 2010), (DICKSON et al., 2010) and (HALAWA et al., 2011)). However, the majority of the capture tools only records video/audio streams and generates, as a result, a single video/audio stream, as a lecture video or a podcast. In several scenarios, this may not be always enough to reproduce the classroom experience.

The classroom itself can be viewed as a rich multimedia environment where audiovisual information is combined with annotating activities (ABOWD et al., 1999). Furthermore, the context of the class (e.g. the slide that is being presented, what the lecturer is saying, where he is looking, his body language, his face expressions, etc.) and how the different audio-visual contents relate to each other are also important. For instance, sometimes it is necessary to relate the slide presentation with the whiteboard for the comprehension of an exercise or lesson (DICKSON et al., 2012). In addition, the interaction between the lecturer and the students is also a valuable part of the learning process.

Such classroom experience and context are usually lost in a captured lecture. In this paper, we propose a way to capture and retrieve lectures and aim to minimize such loss and to offer novel interaction possibilities, not possible in conventional video lectures or when the captured lecture is being "watched". This is accomplished by specifying and capturing the relevant context information to the lecture as much as possible and by producing an interactive multi-video object that is made available for the students. From the multi-video object, the lecture may be reconstituted and explored in dimensions not achievable in the classroom. The student may be able, for example, to get multiple synchronized audiovisual content that includes the slide presentation, the whiteboard content, video streams with focus on the lecturer's face or the lecturer's full body, or the lecturer's web browsing, among others. The student can have the option to choose, at any time, what content is more important to be exhibited in full screen. The student may also be able to perform semantic browsing using points of interest like slides transitions, spoken keywords, lecturer's interactions, etc. Moreover, facilities can be provided for users to annotate the captured lecture while watching it, as suggested by the watch-and-comment paradigm (CATTELAN et al., 2008).

The remaining of this paper is organized as follows: in **Section 2.2** we discuss related works; in **Section 2.3** we describes our proposed model to preserve educational presentations, discussing decisions that guided our design; in **Section 2.4** we present our current prototype implementation and describe a multi-video object generated by a live presentation; in **Section 2.5** we discuss case studies in which instructors used the prototype to capture real classes; and in **Section 2.6** we presents the conclusions and future works.

2.2 Related Work

Several authors report results from building systems designed to capture lectures. The AutoAuditorium, proposed by Bianchi (2004), captures lecture recording classroom activities using a spotting and a tracking camera controlled by computers. The camera orchestration is carried out in real-time using some heuristics based on audiovisual production. The main idea is to create a "TV-like" production without the usual cameraman, video director, audio engineer and other professionals.

The work of Lampi, Kopf e Effelsberg (2008) considers the use of multiple cameras to record lectures. The authors use sensors and computational vision techniques to do the cameraman's job. They also use a finite state machine to define, at each moment, which camera stream should be included in the final stream.

Nagai (2009) uses an environment with a high definition camera (AVCHD) placed at the back of the classroom. This AVCHD can record the whole lecture scene (lecturer, whiteboard, slide presentation, students, etc.). By using tracking techniques, the camera performs digital zoom to the focus of attention.

Chou et al. (2010) use tracking techniques to detect the lecturer's movements and screens (whiteboard, slide presentation) changes. A camera action table is then queried to get what must be done (zoom in, zoom out, pane, etc.) in order to highlight the image that must be the focus of attention.

All the afore mentioned works ((BIANCHI, 2004), (CHOU et al., 2010), (LAMPI; KOPF; EFFELSBERG, 2008), and (NAGAI, 2009)) differ from the work reported in this paper in that the resulting product of the lecture capturing process is just a single video stream, not a multi-video object.

In the work of Liu e Kender (2004), lectures are captured in a similar process to the ones mentioned before, resulting a single video stream. The difference is that the set of slides used in the presentation is added to the video stream. However, the slides are not synchronized with the video. Given that the result is single-video-stream, students do not have autonomy to choose the camera that gives them the best view of the lecture for each situation, or to focus their point of interest.

ClassX is a tool designed for online lecture delivery proposed by Halawa et al. (2011) and Pang et al. (2011). A live lecture is captured by means of an AVCHD stream split in several virtual standard resolution cameras. By using tracking techniques, the most appropriated virtual camera for a given moment is chosen and streamed to the remote students. The students also have the opportunity to choose a different stream from another virtual camera or even watch the original AVCHD stream. A synchronized slide presentation is also offered, but it does not offer navigation facilities to the students.

Schulte, Wunden e Brunner (2008) propose REPLAY, a system for producing, manipulating and sharing lecture videos. REPLAY is similar to the works presented before. Two differences must be pointed out: the use of computational vision to recognize written words and MPEG-7 to index the videos. Although REPLAY allows a more semantic navigation, it does not produce a multi-video object.

The works of Dickson et al. (2012), Dickson et al. (2010), Brotherton e Abowd (2004) and Cattelan, Baldochi e Pimentel (2003) present more advanced features for capturing context information based on image processing and audio transcription. In these works, hypermedia documents are generated with interfaces that provide different ways of indexing the recorded information.

The model for capturing and recovering lectures presented in this paper allows more flexibility. This flexibility comes from the ability to specify the context information that must be captured, and to specify how this context information should be combined to generate a multi-video object (the lecture as a learning object) or to promote live interventions in the classroom during the capture process — for example in the case that there is a change in the illumination of the room (e.g., the instructor turns the light off).

2.3 Ubiquitous Capture: Description and Design

In order to produce quality lecture videos, the conventional lecture recording process usually requires the presence of audiovisual professionals, such as cameramen to operate the cameras, a video director to decide the best frame or camera, and an editor to generate the final version of the lecture video. These professionals increase the operational cost of producing a lecture video and may prevent the wide adoption of lecture capturing. Furthermore, their influence in the lecture may add noise to the communication process between the instructor and the students, making the resulting learning object less effective. Our proposal works toward a self-service approach, allowing the lecturer to record a lecture himself.

Some solutions usually rely on computational vision, tracking techniques and sensors to perform camera orchestrations in a attempt to produce a single video or audio stream output. The model we propose goes a step further, as depicted in Figure 2.1. It captures the lecture in its entirety, capturing all the content presented in the classroom. The capture process is pervasive, does not rely on human mediations and generates automatically an interactive multi-video object which tries to preserve the lecture content and context, as much as possible.

It uses an environment (usually a classroom) which is instrumented with physical devices (Figure 2.1(1)), such as video cameras, microphones, whiteboards, interactive whiteboards and slide projectors. The intrumentalized classroom may also contain sensors, such as temperature sensors and luminosity sensors, and secondary screens, such as notebooks, TVs, tablets, etc. The video cameras should be placed in points where they can frame importants classroom's points (lecturer, students, whiteboard, slide presentation, etc.).

Computer devices capture all the content produced by the physical devices used in the classroom (e.g. whiteboards and slides) and represent them as video, audio and data streams (Figure 2.1(2)). Cameras produce video and audio streams, microphones produce audio streams and sensors produce data streams. By capturing the screen output from the secondary screens or by intercepting the signal sent to the slide projector, we can also produce video streams. The electronic whiteboard can produce both data and video streams. By capturing its strokes we can generate a data stream; intercepting the signal sent to its projector, we can generate a video stream.

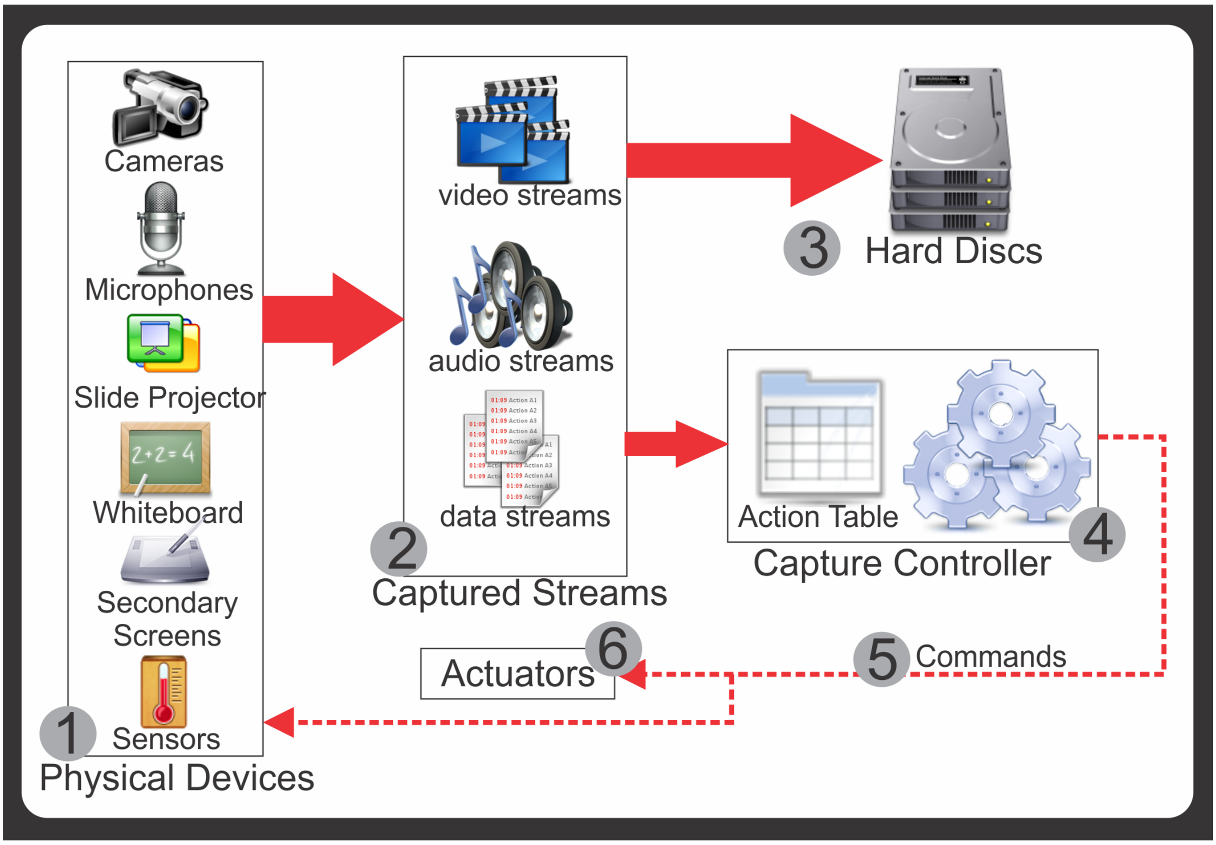


Figura 2.1: Capture Workflow

All such streams are stored (Figure 2.1(3)) for further use in the multi-video object generation. The streams are also sent to the *capture controller* (Figure 2.1(4)), a component responsible for managing the capture process. The *capture controller* uses signal analysis to analyze the captured streams and to send commands (Figure 2.1(5)) back to the physical devices and actuators (Figure 2.1(6)) present in the classroom.

The instructions in the *capture controller* are defined in a customizable action table. The action table can be used to define actions for certain events which may occur during the capture process. For instance, zooming into the image of a specific camera when the lecturer starts talking, or activating an actuator in order to reduce the light intensity when the lecturer starts a slide presentation.

Aiming at a flexible model capable of meeting different educational settings, the instrumented environment should be a multi-purpose room, with support to in-classroom activity and videoconferences. This allows capturing in-classroom lectures with real students (useful in courses with in-classroom activities), videoconference lectures with remote students (useful in distance education courses with synch sessions), and even presentations without students (useful to authoring multi-video learning objects for diffrent educational settings).

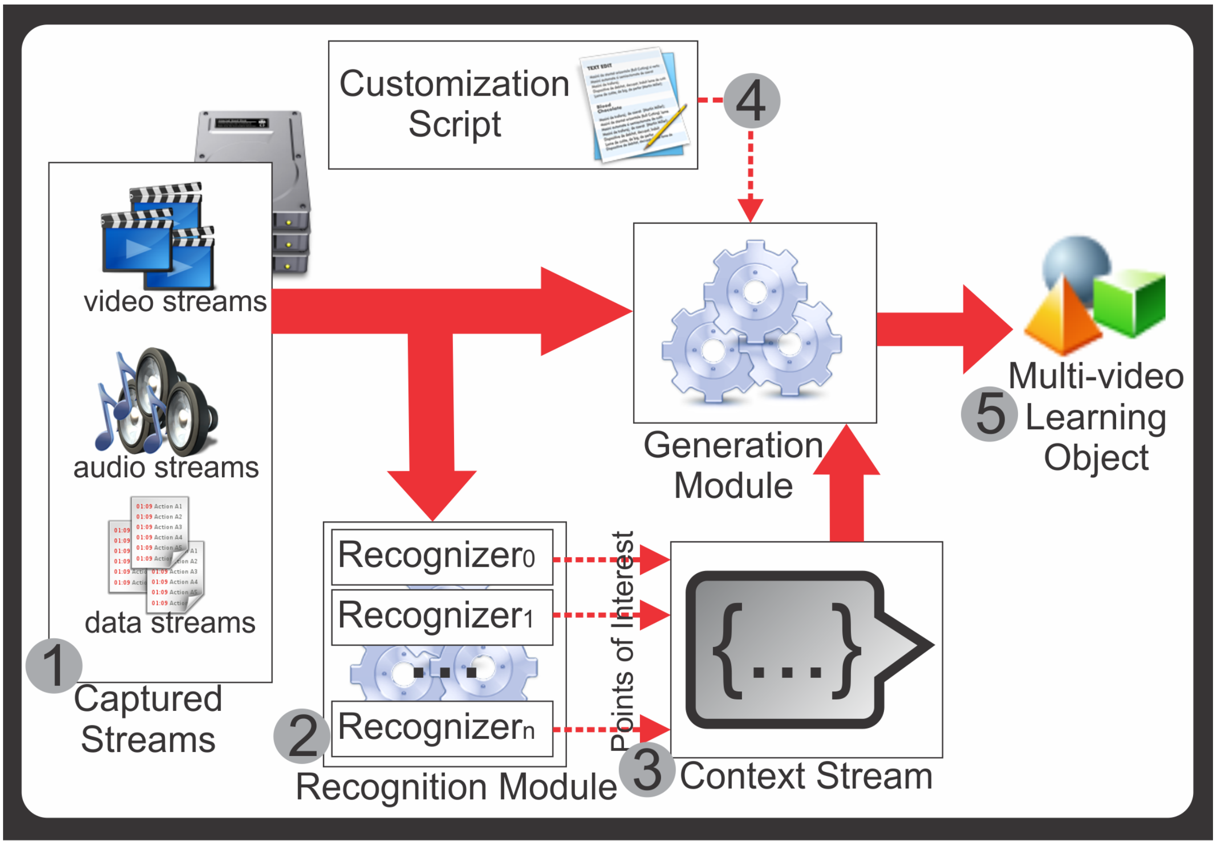


Figura 2.2: Multi-video object generation

Our model allows the instructor to split his presentation in different modules. A multi-video presentation can be composed of one or more modules. This is useful to better organize the content of a lecture. The lecturer may, for instance, prepare a problem solving presentation with one exercise per module. And, it also allows the lecturer to take breaks during the recording process and the students to navigate in the modules of the multi-video presentation.

Splitting the presentation into modules can also minimize the time need for repeating the recording in case of errors. For instance, if the lecturer starts stuttering or becoming confused and wishes to make a retake, he only needs to re-record that module with problems. Reusing the modules to compose a new presentation is another advantage of splitting the recording process into modules — reuse is in fact one of the main ideas underlying learning objects.

The analysis process and the video conversion process of the captured stream can demand much computational power and time. At the end of the capture process the data are transferred to a server for this task. The video, audio and data streams (Figure 2.2(1)) are sent to the *Recognition Module* (Figure 2.2(2)). The *Recognition Module* is composed of several recognizer components and each of them uses one or more captured streams to detect *points of interest*. Points of interest are moments during the classroom which may carry some context information to the students, such as a slide transition or a keyword spoken by the lecturer.

Since many types of points of interest can be conceived using different sensors, devices and other computing analysis techniques, the *recognition module* must have a known interface to allow developers to add new recognizers components as needed.

By composing all the recognized points of interest, the *recognition module* generates the *Context Stream* (Figure 2.2(3)). The *context stream* holds information about what and when something happened during the lecture.

The *Generation Module* consumes the audio, video, data, and context streams to generate a multi-video object. Content and format customizations of the multi-video object may be necessary in order to best suit (i) a course profile (e.g. a course which only uses a whiteboard does not need a slide presentation stream), (ii) the lecturer's style (e.g. if the lecture says "for instance" too often, it may not be a good keyword choice) and (iii) the students' particular needs (e.g. some contrast adaptation for students with visual impairments). The customization can be defined by a *Customization Script* (Figure 2.2(4)) which is used as input by the *Generation Module*. The rules to customize the multi-video object can be described in a declarative language similar to Business Modeling Language (CORALLO; CAPUTO; CISTERNINO, 2007).

The multi-video Learning object (Figure 2.2(5)) is composed of videos and other captured media. Although the multi-video object is not able to reproduce some live lecture experience (live interactions, odors, temperature, etc.), it offers other facilities to the students when they are interacting with the object. These may offset the losses from not having the students present in the classroom when direct interaction between the lecturer and the students can bring benefits to the learning process.

2.3.1 Points of Interest

Points of Interest are moments in the lecture which may be meaningful for students. They may represent changes in the lecture context or moments of more or less importance for the lesson comprehension. The points of interest can be used to provide a more semantic, and even a pragmatic, navigation on the lecture, allowing the students to seek for the next slide transition, for instance.

Some points of interest have been suggested in the literature ((DICKSON et al., 2012), (CATTELAN; BALDOCHI; PIMENTEL, 2003) and (BROTHERTON; ABOWD, 2004)), while others were inspired on our own obseverion of real lectures. Points of interest include:

- **Spoken keywords**: words that denote the context of a lecture. They may be generic as “important”, “for instance"or “exercise”, or lesson-dependent as “Linked List” or “Shakespeare”.
- **Slide transition**: usually denotes a change in the explanation context.
- **Whiteboard interaction**: can be an “writing” or “erasing” activity and can denote the steps of an explanation.
- **Secondary device interaction**: usually denotes steps in an explanation.
- **Change in voice intonation and speech speed**: can denote that some lessons are important or complicated.
- **Students’ intervention**: can denote a meaningful question or addition to an explanation.
- **Lecturer gestures**: can denote something is important or colate two different contents such as whiteboard and slide.
- **Lecturer talking directly to students**: usually denotes a point that the lecturer wishes to emphasize.

2.4 Prototype

As a proof-of-concept of the model presented in **Section 2.3**, we developed a prototype tool for capturing lectures and generating multi-video objects. This prototype was mainly developed in Python. Figure 2.3 depicts an overview of the prototype.

The prototype is composed of three main parts: the *Capturing tool* used to capture streams; the *Processing tool* in charge of stream analysis and the generation of the multi-video object; and the *Presentation tool*, which allows the user to playback the multi-video object.

2.4.1 Capturing tool

The *Capturing tool*, named *Classrec*, (Figure 2.3(1)) performs the lecture capturing process. Each computer used in the capturing process runs an instance of *Classrec*, and one of these instances is selected to be the session manager (Figure 2.3(2)). It corresponds to the *Capture Controller* of the capture workflow (Figure 2.1). The session manager is responsible for handling the lecturer’s stimulus and for controlling the other *Classrec* instances, keeping they synchronized.

The capturing process is based on video streams. *Classrec* captures content (video and audio streams) produced by AVCHD and outputs produced by computers (such as computer

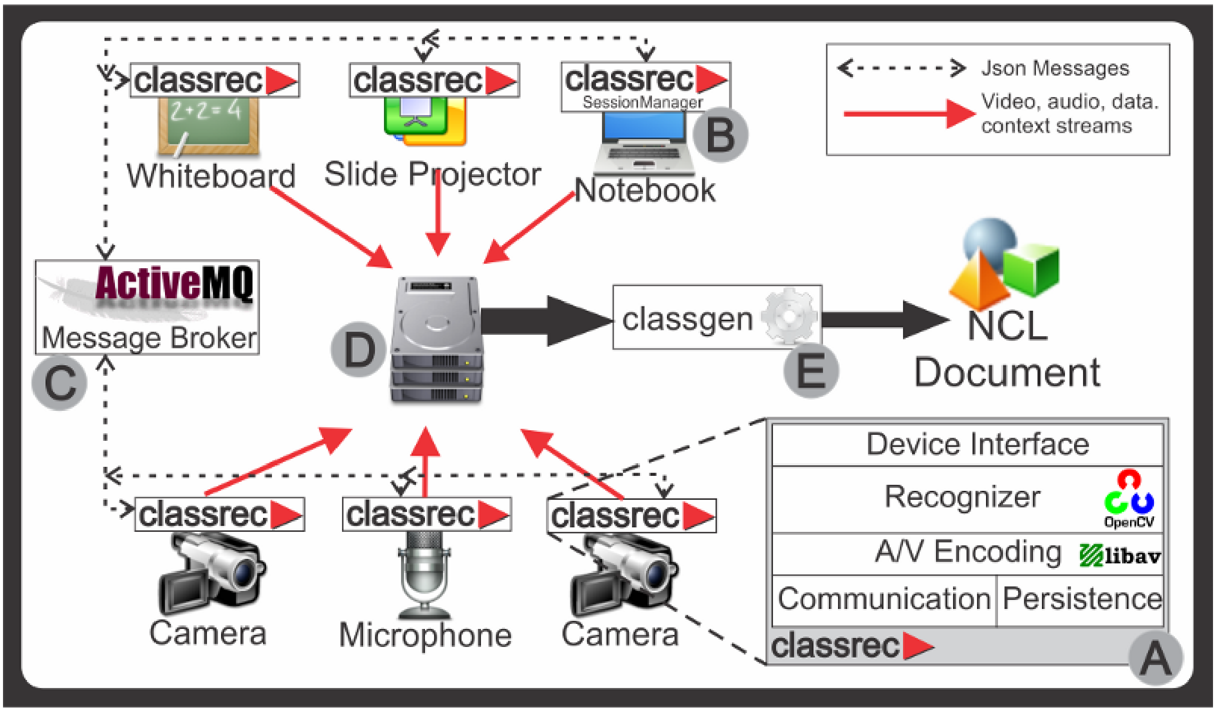


Figura 2.3: Prototype Overview

screens, slide presentations, etc.).

We opted to capture the electronic whiteboard output as a video stream instead of its strokes. This was done because a video stream is more portable than strokes and, given the modern video encoding as h.264 Advanced Video Codec and the static nature of whiteboard outputs, the bit rate of the video stream is low. We could record a stroke stream, but it would require a specialized media player to play it back (as it is the case with other systems).

Some streams, such as slides, whiteboards, computer screens, can have segments with a lot of static content, but they are still captured as video streams. A possible improvement would be to replace the video for a combination of non-static content videos and a single image to represent a static segment (video with no changes during a period of time).

Classrec records audio/video streams using the *libav* library[1]. It also records metadata about the lecture, such as module structure, available streams and authoring information in an XML file, named *XML lecture*.

The communication among the different applications is carried out using the Apache ActiveMQ message broker (Figure 2.3(3)).

[1] Libav - http://libav.org/

2.4.2 Processing tool

The *Processing tool*, named *Classgen* (Figure 2.3(4)), performs the multi-video generation process. This tool uses as input the video streams and metadata recorded by *Capturing tool.* It also supports an XML configuration description language, which allows the specification of which recognizers (and its inputs) should be used, and the codecs that should be used to encode audio and video.

We have implemented recognizers capable of detecting (i) the presence of a lecturer in a video steram; (ii) if the lecturer is facing a camera; (iii) slides transitions; (iv) interactions with whiteboard or PC; and (v) some spoken keywords.

It is also possible to specify an orchestration of video streams in order to produce a new video stream. This is useful in environments with multiple cameras recording different angles of the lecturer. Through the XML configuration description language, it is possible to select which stream will be used in the orchestration and how to orchestrate then. For instance, it is possible to specify that when a recognizer detects the lecturer's face in video segments, the camera orchestration stream should include that segment.

Classgen uses the OpenCV library(BRADSKI, 2000) to perform pattern recognitions in order to identify points of interest for composing the context stream. The media manipulation during the orchestration process and the audio/video conversion is handled by the libav library.

After performing the recognition, orchestration and video conversion, the information generated by these processes (the points of interest, the orchestration stream, the converted streams) are stored in the *XML lecture*. The XML is then passed to a component of the *Processing tool* responsible for generating the final multi-video object (Figure 2.3(5)). Our prototype generates NCL[2] documents, but the *Classgen* can be extended to generate other types of multi-video objects, such as HTML5 pages or stand-alone desktop, tablet or smartphone applications.

The XML configuration description language can also describe the video streams (including the orchestration, if any) and points of interest will be used in the final multi-video object. It is also possible to generate different multi-video objects using the same recorded lecture (for instance, by using the orchestration stream or not).

[2] Nested Context Language - http://ncl.org.br/en

2.4.3 Presenting tool

It is desirable to offer students a platform-independent way to access the captured lectures. We would like to avoid students having to install specific software to playback the lecturers. To fulfill this requirement, two possibilities were considered:

1. Generate an HTML5-based multi-video object, to be presented in standard web browsers, using JavaScript code to perform media synchronization.
2. Generate a multi-video object based on a synchronization language as NCL or SMIL (BULTERMAN; RUTLEDGE, 2008).

The multi-video object generated from the capture imposed some challenges. In the scenario where we have considered the generation of the object directly in HTML5 + JavaScript, a large development effort to implement the synchronization capabilities is demanded. We also noticed that most obstacles identified in the HTML5-based implementation would be easily overcome with the use of a declarative language specialized in media synchronization. However, there were no solutions to support it that did not demand external plug-ins.

As a result of these needs, we were motivated to propose and develop a multimedia presentation engine based on standard Web technologies. We conducted an implementation based on HTML5 + JavaScript that enables the presentation of multi-video NCL documents, named WebNCL[3] (MELO et al., 2012). Thanks to WebNCL, any device which has an HTML5-compatible browser (PC, Smart TV, Tablet, Smart Phone, etc.) can present NCL documents natively.

The choice for implementing support to the NCL language was taken because it is a powerful language for media synchronization, under active development and adopted as iDTV (ABNT, 2007) and Internet Protocol TV (IPTV) standards (H.761, 2009; MORENO; BATISTA; SOARES, 2010). A good side effect of this choice was the possibility to reuse the content generated in different platforms.

Figure 2.4 shows some running NCL learning objects generated by the prototype. The NCL document offers some facilities for students. One of these facilities is the synchronization of the captured audio/video. The multi-video object synchronizes the multiple audio/video streams, so students can see what was written in the whiteboard when the lecturer points to the slide presentation. This synchronization is essential to recover the whole audiovisual context of the

[3] WebNCL is an open-source software, available at http://webncl.org

captured lecture at a given moment. It is also possible to insert non-synchronized complementary media to the multi-video object like, for instance, an image from a textbook.

The multi-video object offers a more semantic and easy way to navigate in the captured lecture than timeline navigation, common in video (however, timeline navigation is still present). For instance, the student can move forward to the next slide transition or backwards to the previous one. When the lecturer begins to write something in the whiteboard, the student can skip all the writing process and see the final result. In a future implementation, students will also search for a keyword and move forward in the multi-video object to the point where the lecturer said “for instance”.

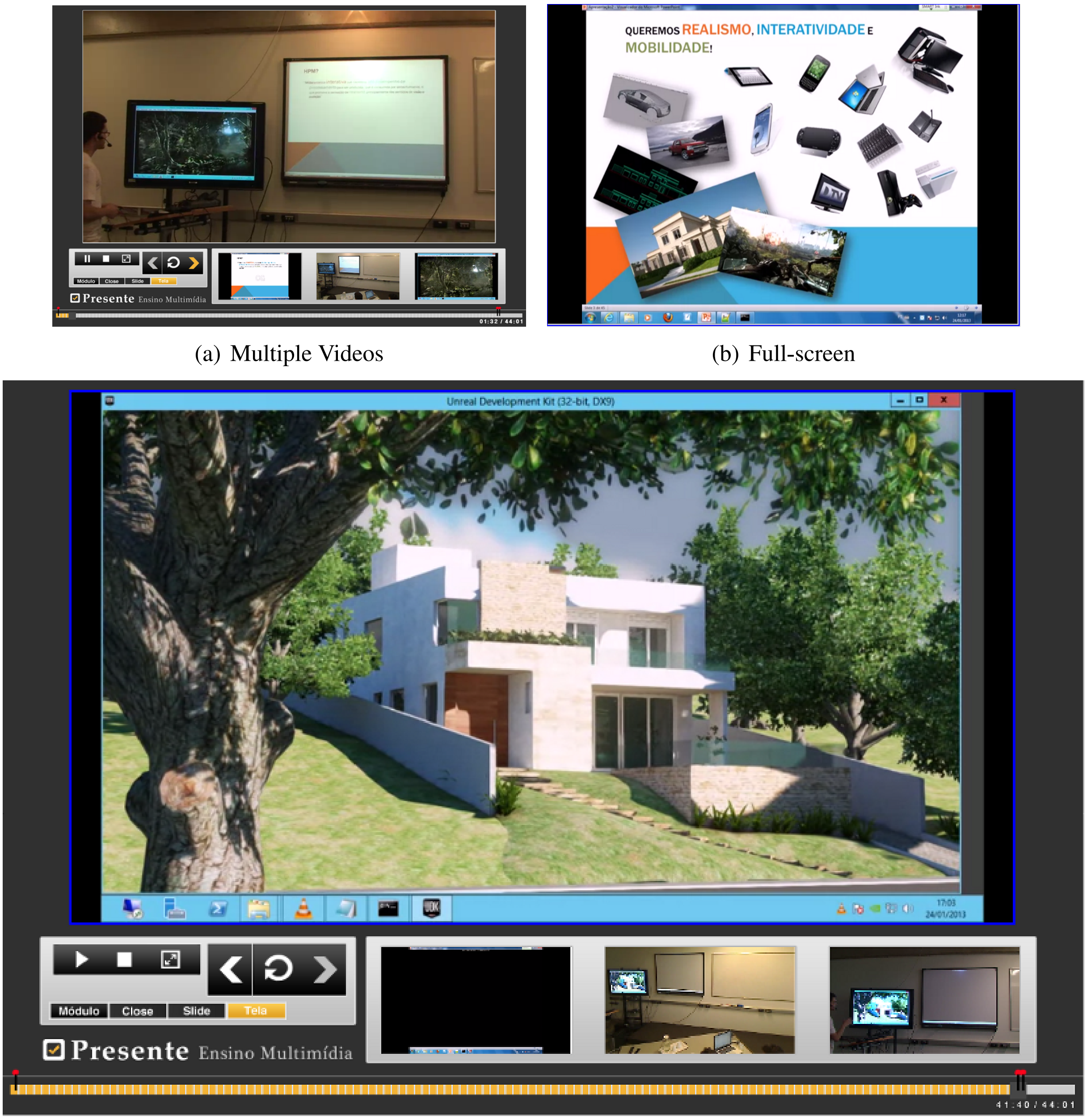
Similar to in-classroom lecture, wherein the student can pay attention to different spots (the lecturer, whiteboard, slide presentation, the textbook, or another screen), the multi-video object allows the student to choose which video stream he wishes to see and he can even see more than one at the same time.

Finally, the student has the facility to make annotations in the multimedia object by means of the watch-and-comment paradigm. For instance, he can mark some part of the lecture as important or irrelevant, or he can delimit a snippet of the lecture which he did not understand for further research or to ask the professor or tutor. He can also make comments on the lecture via audio or text, in similar in-classroom students do with paper and pencil.

2.5 Case Study

The capture-tool prototype was deployed in a multi-purpose room at the Computing Department of Federal University of São Carlos. We invited six professors to use the prototype and record presentations. Four of them recorded a lecture simulation (without students), one professor a conventional lecture, with real students and one recorded a problem solving class. We also used the prototype to record a term paper presentation and a conference presentation.

Figure 2.5 shows photos of the instrumented room. At the front (Figure 2.5(a)) there is a conventional whiteboard, an electronic whiteboard and a notebook in which the presenter can browse the web or use any other software. The interactive whiteboard can be used to present slides (there is a Bluetooth presenter to control the presentation) and it allows drawing and writing over the screen. At the back side (Figure 2.5(b)), we placed two AVCHD, each one framing the interactive or the conventional whiteboard. We also placed a webcam as a wide-shot cam, framing the whole front of the room. As the room is a shared space in the university, we needed to lock the equipment in cabinets.

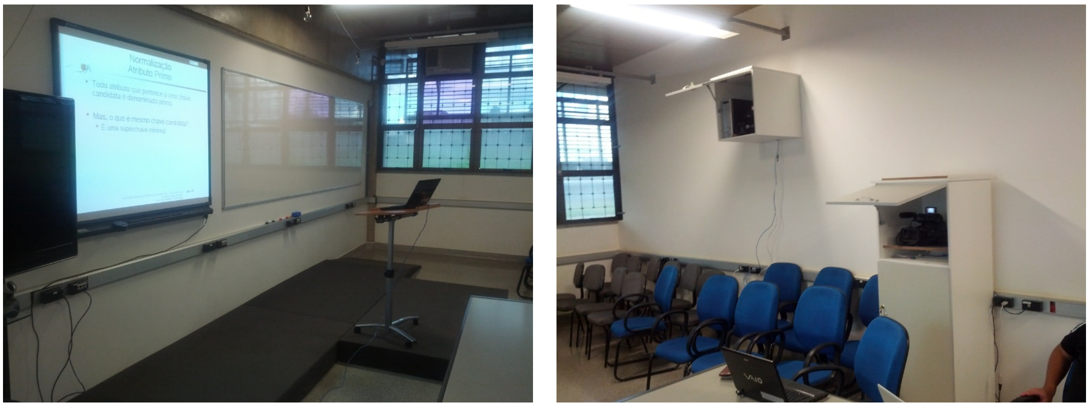

Módulo
Close
Slide
Tela
☑ Presente Ensino Multimídia
01:32 / 44:01
QUEREMOS REALISMO, INTERATIVIDADE E MOBILIDADE!
Unreal Development Kit (32-bit, DX9)
17:03
24/01/2013
41:40 / 44:01

(a) Multiple Videos

(b) Full-screen

(c) Timeline

Figura 2.4: Multi-video learning objects



(a) Front Side (b) Back Side

Figura 2.5: Instrumentalized Room

After a short explanation of how to use the capture tool, all lecturers could perform the recording alone, meeting the proposed self-service approach. Since the room is a real classroom, the lecturers could perform their presentations naturally, despite the cameras. Most of them did not modularize the presentation and record a single long module. We also observed that most lecturers did not use all the resources available in the environment.

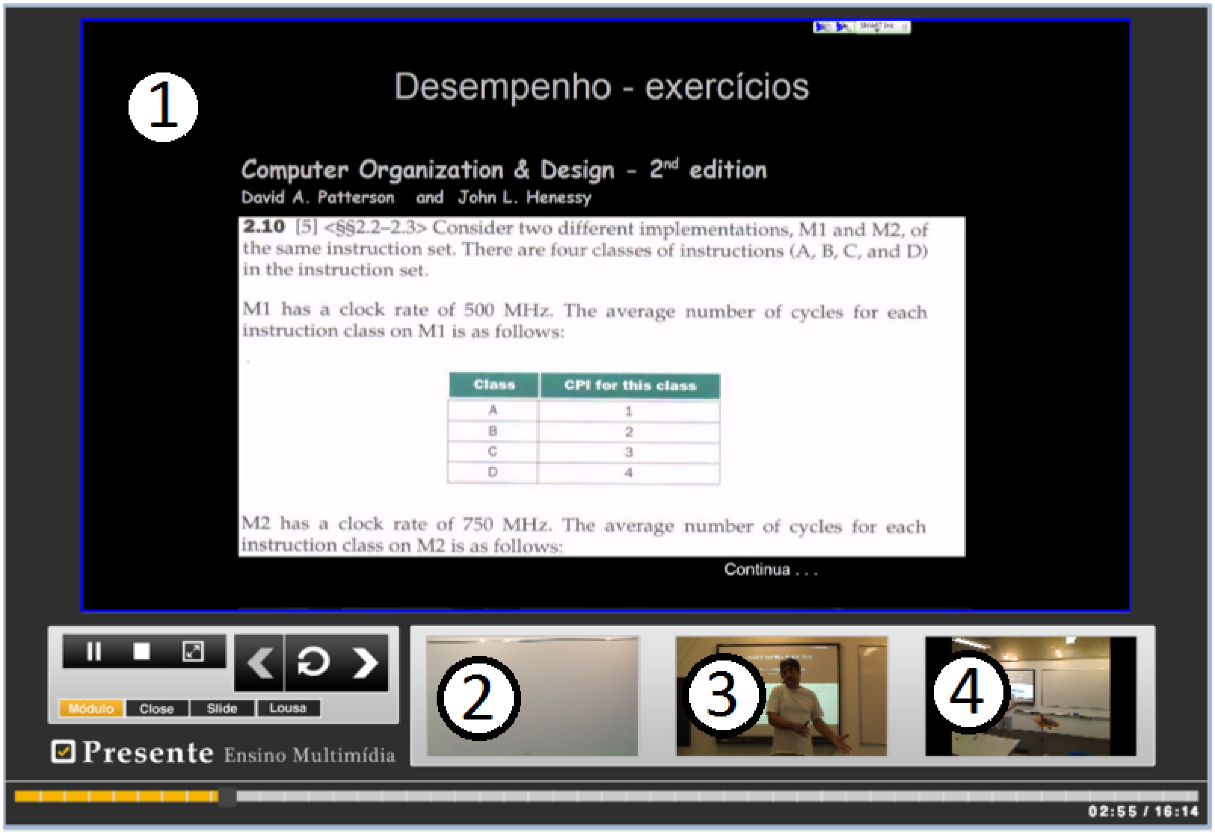
After the capturing process, we carried out interviews with the lecturers. The testimonies suggest the needing of a guideline for how to prepare presentations that better exploit the resources. The testimonies also indicate that lecturers enjoyed the experience and found the tool useful.

2.5.1 Playback Metrics

We selected one of the captured presentations to perform a study case with real students. The presentation was a problem solving session for a Computer Organization course. It was organized into 12 modules, in a total of 1 hour and 18 minutes of content.

Figure 2.6 depicts the multi-video object generated from the presentation. There are 4 streams: (1) slide projection capture; (2) camera focused on the conventional whiteboard; (3) camera focused on the slide presentation; and (4) a wide-shot camera. Although the generation process has the camera orchestration feature, in this study case we did not use it. The aim was to exploit the students' interaction, forcing them to choose, for a better experience, which would be the video to be presented in the main (bigger) window at each instant .

The multi-video object was placed in the web and, using the log API of WebNCL, we logged all the interactions carried out by the students. Eighteen students watched the presentation for at least 4 minutes. The students' average playback time is 3542.67 seconds (about 59 mi-



Desempenho - exercícios

Computer Organization & Design - 2nd edition
David A. Patterson and John L. Henessy

2.10 [5] <§§2.2–2.3> Consider two different implementations, M1 and M2, of the same instruction set. There are four classes of instructions (A, B, C, and D) in the instruction set.

M1 has a clock rate of 500 MHz. The average number of cycles for each instruction class on M1 is as follows:

Class	CPI for this class
A	1
B	2
C	3
D	4

M2 has a clock rate of 750 MHz. The average number of cycles for each instruction class on M2 is as follows:

Continua . . .

Figura 2.6: Problem Solving Presentation

nutes) with a standard deviation of 2382.23 seconds (about 39 minutes). The average number of students' interactions (play, pause, navigations, selection of the main video) is 118.55 with a standard deviation of 99.58.

Figure 2.7 summarizes the watching attendance of module 1 from the presentation. The horizontal axis is the number of seconds of the module 1 (which is 974 seconds). The blue line represents the number of times the instant was watched by students, and the red line the number of different students watched that instant.

Since the module always starts from 0 seconds, it is natural that the attendance of the first seconds is bigger. The points where the blue line is above the red line mean that the moment was watched more than once by the same students. This graphic can be useful for lecturers to find out which parts of a lecture are more useful or important for the students. For instance, around the second 213 there is a local maximum point, which is the same moment a slide transition occurs in the multi-video object.

Figure 2.8 summarizes which streams were more selected as the main stream in each moment of module 1. Each line represents how many times a stream was watched in a specific moment. The blue line refers to the slide projection capture (Figure 2.6(1)); the red refers to the

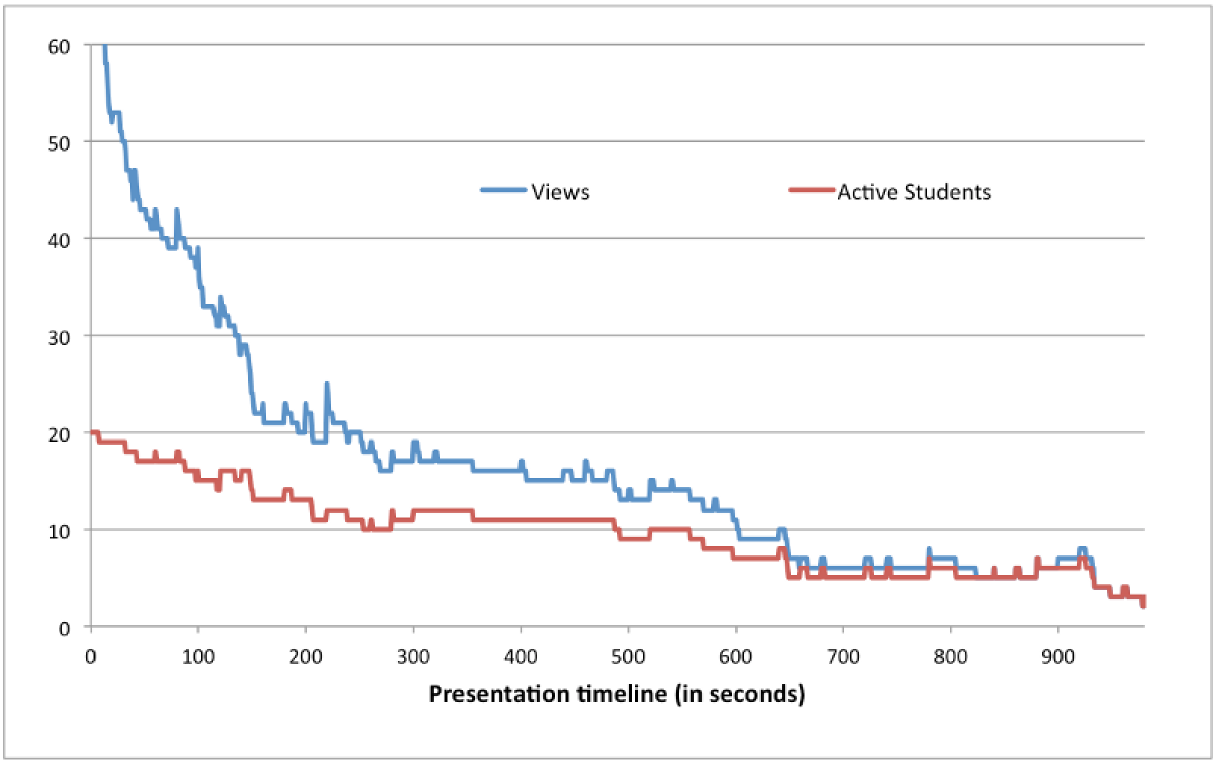


Figura 2.7: Playback Views

camera focused on the conventional whiteboard (Figure 2.6(2)); the green camera focused on the slide presentation (Figure 2.6(3)); and the purple the wide-shot camera (Figure 2.6(4)).

According to Figure 2.8, the more watched streams were the slide presentation and the whiteboard camera. We can also note that the slide presentation are more watched near the moments when there are slide transitions (seconds 213 and 515).

Figure 2.9 depicts the behavior of 2 students when interacting with the presentation. The horizontal axis is the playback timeline and the vertical axis is the presentation timeline. **Student A** (blue line) spent about 1800 seconds (30 minutes) in the the module 1, whereas **student B** (red line) spent about 1200 seconds (20 minutes). Vertical straight lines (such as Figurer 2.9(1) and Figure 2.9(2)) represent a navigation that the student performed during playback.

We can note that **student B** starts watching from second 180 and performed some backward moves mainly in the end of the presentation. **Student A** watched almost linearly until second 650 and then returned to the beginning of the presentation and watched it again until the end.

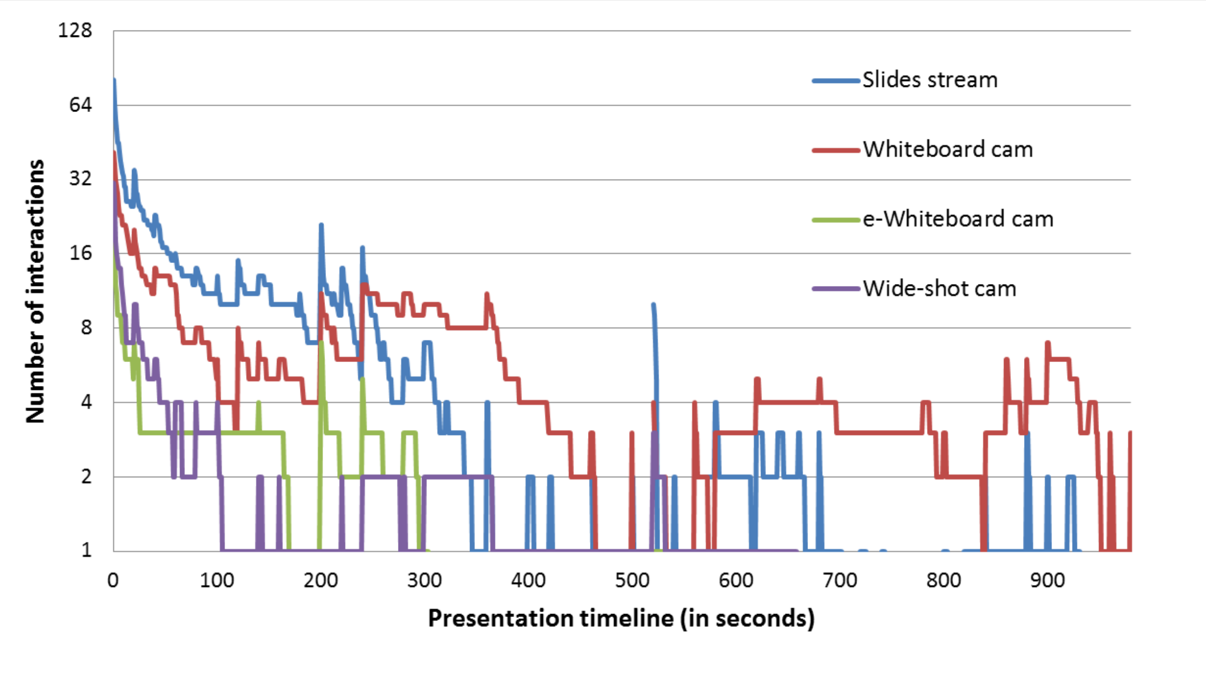


Figura 2.8: Streams Views

2.6 Final Remarks

This paper suggests the preservation of multimodal and multi-device experience of educational presentations. Capturing both the content presented on multiple devices and important context information from different angles allow the generation of an interactive multi-video document that tries to preserve the level of reality of the presentation as much as possible. From the multi-video it is possible to reconstitute and explore the lecture in dimensions not achievable in the classroom. This may offset the losses from not having the students present in the classroom when direct interaction between the lecturer and the students can bring benefits to the learning process.

Other reasons to preserve the reality of educational presentations that deserve to be highlighted again involve contributing with the following actions: reusing the presentation, when the same subject needs to be presented to a different audience; refining the presentation making use of the feedback provided by students, explicitly through their comments and annotations and implicitly through the log of their navigation over the multi-video object; planning —- the same explicit or implicit student's feedback can help the lecturer or coordinator to reorganize the course; providing a mechanism for student evaluations, probably not for grading score; serving as a learning reinforcement for students who want to review the subject; to serve as an educational object in self-study or in a distance education course; among others.

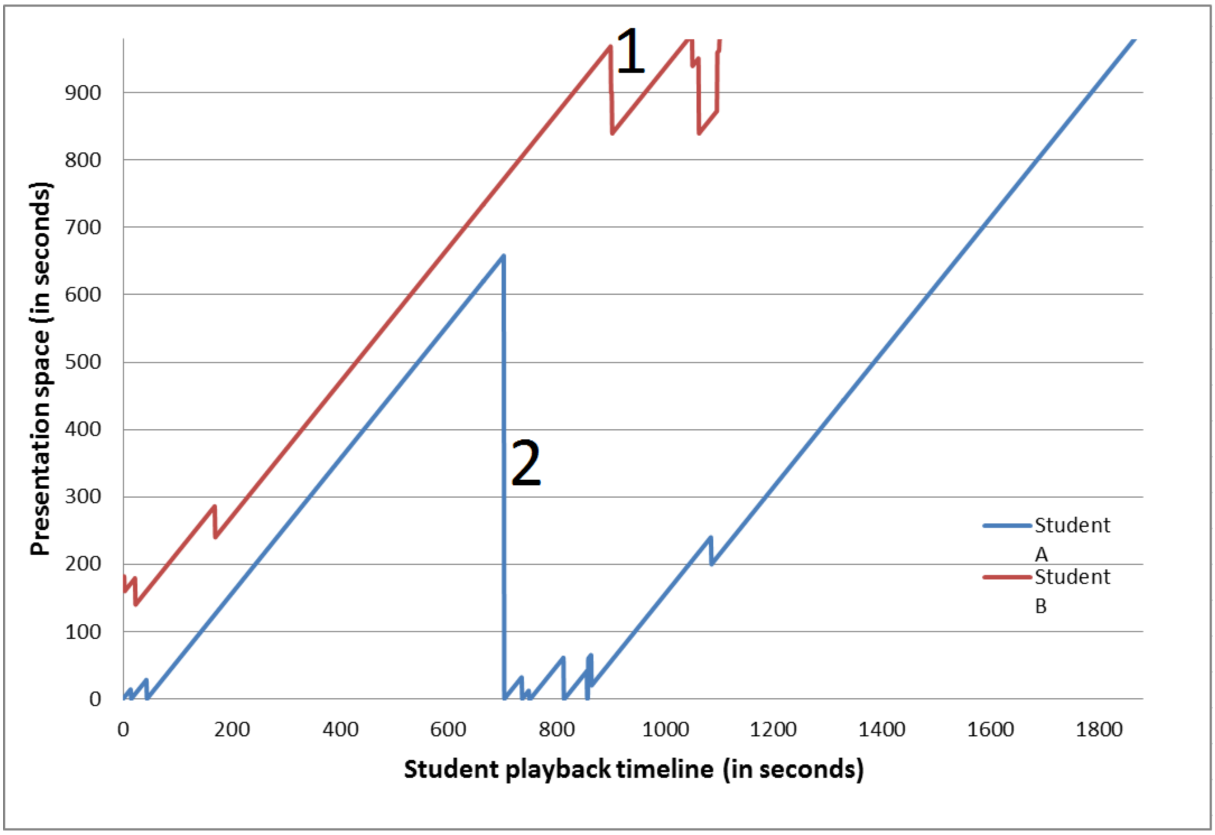


Figura 2.9: Students Navigation

The construction of an infrastructure to capture different views of an educational presentation in order to generate an equivalent multi-video object and to present it, allowed testing concepts with faculties and students and identifying some improvements and adjustments that must be made.

The graphs in Section 2.5.1 were only presented in order to show the type of data that can be collected during the playback of (navigation over) a presentation and how this data can be transformed into useful information for faculties, students, researchers, etc. There was no intention to perform data analysis. In another paper (VIEL et al., 2013b) the authors detail several other forms to present such information and, together with those responsible for the content featured, perform some data analysis.

Analysis tools must be built to facilitate the extraction of useful information from the large amount of data generated by students navigating over multi-video presentations. Research on education that explore the impact of students using multi-video as learning objects, may be of interest. The solution presented here is not restricted to educational presentations. Other areas such as business, religious, social, artistic, etc., can also be contemplated.

Capítulo 3

WebNCL: A Web-based Presentation Machine for Multimedia Documents

Presentation machines for multimedia declarative languages – especially the ones related with Interactive Digital TV (iDTV) and Internet Protocol TV (IPTV) – are usually embedded in devices and strongly coupled with the platforms when native code and API for the device's platform are used. Since much of the complexity to implement presentation machines lies on presenting and controlling different types of media (video, audio, image, text), and given that most of the modern browsers natively support those requirements, it becomes interesting to implement presentation machines using Web technologies to reduce their coupling with platforms. In this paper we discuss the advantages of a presentation machine for declarative multimedia languages implemented on top of Web technologies. As a proof of concept we implemented the WebNCL, a lightweight NCL presentation machine based on the web technologies stack (HTML 5/ JavaScript/ CSS). By using WebNCL, NCL documents can be presented in any device that has a HTML5 compatible browser, such as tablets, smartphones, smart TVs and PCs.

3.1 Introduction

Implementations of presentation machines for multimedia declarative languages – especially the ones related with Interactive Digital TV (iDTV) and Internet Protocol TV (IPTV) – are usually embedded in devices and strongly coupled with the platforms when the native code and API for the device's platform are used. For instance, the iDTV middleware Ginga-NCL (SOARES; RODRIGUES; MORENO, 2007; SOARES; MORENO; NETO, 2010) contains a Nested Context Language (NCL) (SOARES; RODRIGUES, 2006) presentation machine.

The Hypertext Markup Language (HTML) and other languages used to build Web pages, such as JavaScript and Cascading Style Sheets (CSS), are platform independent, because they are interpreted by Web browsers. Given the wide variety of browsers present in many platforms (PCs, smartphones, tablets, smartTVs, videogame consoles), it is becoming common to implement applications on top of the Web technologies to make them platform independent.

Since much of the complexity to implement presentation machines lies on presenting and controlling different types of media (video, audio, image, text), and given that most of the modern browsers natively support these requirements, it may be interesting to implement presentation machines using Web technologies to reduce their coupling with platforms.

Furthermore, there are several advantages in having a presentation machine for multimedia declarative language objects based on standard Web technologies. First it allows the reuse of declarative multimedia presentations developed for specific environment in other platforms. In addition, in a Web development perspective, a presentation machine built on top of web technologies stack (HTML5-JavaScript-CSS) is a framework which offers facilities for creating Web-applications with media synchronization requirements. The integration of multimedia declarative language objects, such as NCL documents into Webpages can enhance the capabilities of web applications and reduce the development costs.

Since a Web-based presentation machines are simpler to be used and integrated with other applications than native implementations, a more flexible and portable toolkit for development (testing and prototyping) of multimedia presentations can be provided. The presentation machine could be integrated with a text editor and provide real-time feedback or allow the presentation of multimedia objects without the necessity of sending into real or emulated devices.

Web browsers are available in many platforms. They can be found in smart TVs, smartphones, tablets and other devices. The device manufacturers can reuse the existing infrastructure provided to web browsers to integrate a multimedia presentation machine to their products. For instance, a web-based presentation machine provides a less invasive solution to add support for declarative applications in STB than embedding a dedicated middleware to present declarative multimedia documents, which usually runs on system space and communicate directly with the operation system, the presentation machine can run as a Web-application in the user space.

Finally, a Web-based presentation machine allows the use of rich declarative multimedia languages, such as NCL, to create classes of presentations that are not common in embedded environments due to platform restrictions.

In this paper we discuss the advantages of a presentation machine for declarative multimedia languages implemented on top of Web technologies. As a proof of concept we have implemented the WebNCL, a NCL presentation machine based on the web technologies stack (HTML5/JavaScript/CSS). By using WebNCL, NCL documents can be presented on any device that has a HTML5 compatible browser, such as tablets, smartphones and PCs.

The remaining of this paper is organized as follows: in Section 3.2 we discuss related works; Section 3.3 describes the development process of the WebNCL and important project decisions; Section 3.4 presents the architecture of the WebNCL; Section 3.5 describes how WebNCL can be integrated into a webpage and presents the compatible platforms; in Section 3.6 we discuss some aspects related to embedding WebNCL; Section 3.7 presents the conclusions and future works.

3.2 Related Work

The integration of web technologies with media synchronization languages has been investigated in some works. Cazenave, Quint e Roisin (2011a) propose a solution to integrate SMIL Timesheets (VUORIMAA; BULTERMAN; CESAR, 2008) into HTML documents. Their solution allows web applications to use the SMIL temporal synchronization features through the engine *Timesheet.js*, implemented in JavaScript. A similar solution was adopted by Vuorimaa (2007). The solutions proposed by these authors include only a subset of SMIL and the focus is on providing synchronization mechanisms to web pages. The creation of full portable multimedia presentation machines was not investigated by them.

Jansen e Bulterman (2008) and Jansen e Bulterman (2009) present a solution for integrating SMIL documents to web pages through the use of browser plugins. A bridge between the Ambulant Annotator (CESAR; BULTERMAN; JANSEN, 2006) and a *webkit*-based browser was created to perform time synchronization in web pages.

Gaggi e Danese (2011) proposes the *SmilingWeb*, a SMIL presentation engine that runs on any browser that supports HTML5. Our work takes this direction and uses NCL as base for discussions and proposals. The implications of this approach on DTV systems and mobile devices are also pointed in our paper.

In the context of Digital TV systems and NCL Language, Cruz, Moreno e Soares (2008) describes a reference implementation of Ginga-NCL for mobile devices based on the Symbian platform. The focus of that study is to validate the Ginga-NCL architecture and to verify how it can be used to port the reference implementation to mobile devices. Ferreira et al. (2010) present an implementation of middleware Ginga-NCL for mobile devices based on Android platform. Their paper describes the steps for porting the middleware to that platform and evaluates the transmission on applications for mobile devices using the Digital Storage Media — Command and Control (DSM-CC) (CRINON, 1997) protocol through IP networks. Although they present the steps to port the middleware, they do not present solutions for cross platform development.

Other Digital TV middlewares have adopted web technologies for their declarative subsystems. This is the case of Broadcast Markup Language (BML) that is based on XHTML, CSS, Document Object Model (DOM) and ECMAScript (ECMASCRIPT; ASSOCIATION et al.,). The proposal described in this paper can solve some of the known problems of these systems, such as restrictions of adaptability, support for multiple devices and live presentation editing. As an example, WebNCL could be integrated to those middlewares, allowing them to display multimedia documents specified in NCL language by relying on technologies already supported (web-technologies).

3.3 Project Decisions

The WebNCL was developed using agile methods. Although a specific method was not used, we used a mix of SCRUM (SCHWABER; BEEDLE, 2001) and XP (BECK; ANDRES, 2004).

We used the NCL ABNT Standards (ABNT, 2007) as our main reference during the development. We also consulted Soares (2009) to clarify some NCL features and concepts which were not clear in the standards. Our previous experience ((PIMENTEL et al., 2010),(FREITAS; TEIXEIRA, 2009),(TEIXEIRA et al., 2010)) modifying the middleware Ginga-NCL reference implementation (SOARES; RODRIGUES; MORENO, 2007) contributed to define the requirements of our presentation machine as well as its architecture.

Before starting the development itself, we conceived some proofs-of-concept to determine the viability of implementation and tested some approaches. For instance, we developed an event manager to test the NCL links condition-action mechanisms. We also performed some stressing-tests playing several videos side by side and generating events in a high rate.

After the viability tests, we have conceived an architecture for the WebNCL presentation machine. At the beginning it was similar to the Ginga-NCL reference implementation architecture (SOARES; RODRIGUES; MORENO, 2007), but it suffered many changes during the process due to the JavaScript language particularities and in order to meet project decisions.

The first goal was to develop essential and core parts of the presentation machine, such as the NCL Document parser, NCL Representation Model and the Event Manager. The remaining parts of the implementation was driven by features. We chose features, such as transition or some descriptor propriety, and added support to it. After that, a simple NCL document was written for testing purposes. That NCL document was executed into the Ginga-NCL Virtual Machine and in our presentation machine running in different browsers (Chrome, Firefox, Opera and Safari) and in different operational systems (Linux, Windows, Mac OS, Android and iOS) to verify the implementation accuracy.

3.3.1 Languages and Platforms

In the project conception phase, we considered implementing our NCL presentation machine based on Adobe Flash Platform[1]. Different from the HTML5 stack, Adobe Flash is already a mature platform. It also has great media support (especially video) including well-proved streaming protocols and technologies. In addition, its video formats are homogeneous – all up-to-date Adobe Flash Players support the same video formats. Another advantage is that by using the Adobe Air[2], we could easily develop a standalone desktop presentation machine.

However, a HTML5-based implementation can be used in more platforms since the HTML5 stack is based on open standards. For instance, iOS devices, such as iPhone, iPad and AppleTV, do not support Adobe Flash, but they have HTML5-compatible browsers.

We also believe that HTML5 is an ascending technology. Although it is only a draft and has heterogeneous support across browsers, it is already being used to build interesting and complex Web Application.

Computer power in presentation machines is required more to media presentation than to media orchestration. Since the web-browsers are responsible for media presentation and are usually implemented in native code, we believe the performance difference between WebNCL and the native-code presentation machine may be irrelevant.

3.3.2 Frameworks

Although HTML5 has native support for playing and controlling videos, we decided to use the Popcorn.js framework[3] for video playback. Porcorn.js is a JavaScript framework developed by the Mozilla Foundation for media synchronization and control; it also has an abstraction layer that allows developers to use HTML5 native videos via <video> tag and Adobe Flash

[1] http://www.adobe.com/products/flashplayer.html

[2] http://www.adobe.com/products/air.html

[3] Popcorn.js - http://popcornjs.org/

video by a Flash Player in an agnostic way.

Popcorn.js offers a higher API for control media than pure HTML5. It is also well-tested in different browsers and platforms, which makes the presentation machine consistent across different browsers. Its abstraction layer can be used to add support for Adobe Flash based videos in our NCL presentation machine. It can be used to add features that cannot yet be used with HTML5 native videos, such as video streaming by Real Time Messaging Protocol (RTMP) [4].

A potential drawback when adopting Porcorn.js is the increased size and execution time of the JavaScript code. The size of a minified version of the framework core is 18,4KB and a version with all official plugins is 143KB. This overhead is not significant considering usual broadband Internet connection speed. We have not observed performance differences when compared a HTML5-pure and a Popcorn.js implementation. Also, the JavaScript code computing cost time is negligible when comparing to the cost of the video playback.

We also adopted the JQuery framework[5] since it makes the JavaScript development simpler and enhances presentation machine portability and reliability across different browsers and platforms.

3.3.3 Time-based Transition Handling

The Ginga-NCL reference implementation uses a scheduler to track the NCL presentations time-based transitions, such as the beginning of an anchor or the end of a video media. This is carried out by a thread that calculates all the presentation transitions and creates a timeline with those transitions. The thread waits the first transition and then executes the appropriated link actions. Sometimes it's necessary to recalculate the transitions timeline because the actions of the links can start or stop medias, changing the order of the transitions.

Given the asynchronous and event-driven characteristics of the JavaScript language, we adopted a different strategy to process time-based transitions in our presentation machine. The responsibility to inform when transitions occur is distributed among the players. Each media player has the information about the time-based transitions associated with its media (anchors). When a transition occurs the player triggers an event to the Event Manager.

If there are links that use a transition as condition, the Event Manager creates a listener for the corresponding event triggered by the player. When an event is triggered, the listener calls

[4] RMTP (Real Time Messaging Protocol) is an Adobe proprietary streaming protocol

[5] JQuery Framework - http://jquery.com

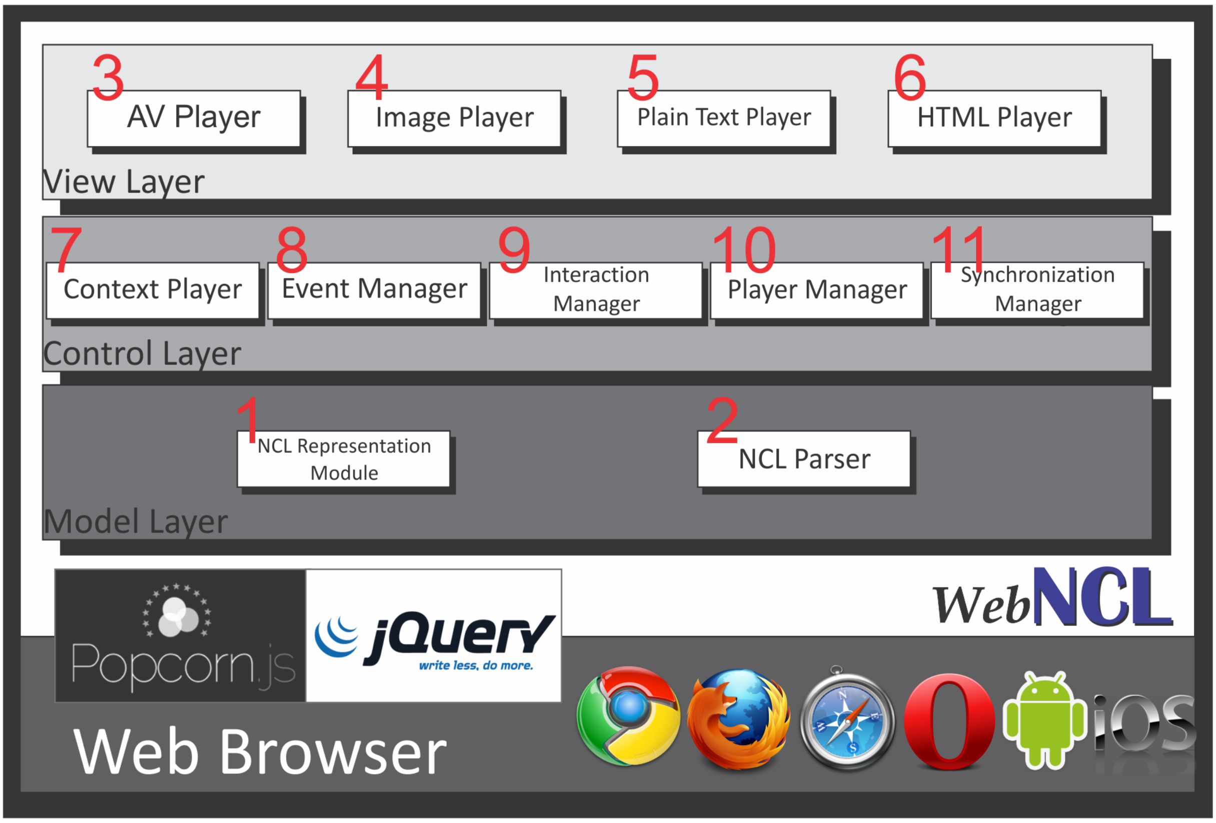


Figura 3.1: WebNCL Architecture

the functions for each link that depends on such a condition. If all the conditions of a link have been met, the link will execute its actions, triggering events to media players. The media players listen to events performed over its medias and execute the corresponding actions.

This approach provides a decoupled architecture, since the player only needs to know the transitions related to its media and the actions that can be performed over it. The Event Manager is agnostic and does not need to know anything about the media nature.

3.4 Architecture

The WebNCL was designed using the Model-View-Controller (MVC) (GAMMA, 1995) design pattern. Figure 3.1 depicts its architecture.

The Model Layer (Figure 3.1 bottom) consists of two components: NCL Representation Model (Figure 3.1 - component 1) and NCL Parser (Figure 3.1 - component 2). The NCL Representation Model contains JavaScript objects to represent each entity of NCL Documents (link, media, connector and etc.) and their values. The NCL Parser is responsible for translating an NCL Document represented in Extensible Markup Language (XML) code into objects of the NCL Representation Model component.

The View Layer (Figure 3.1 top) contains the media players. Those players are implemented in HTML5 and JavaScript code using the Popcorn.js framework as base. There is an Audio and a Video Player (Figure 3.1 - component 3), an Image Player (Figure 3.1 - component 4), a Plain Text Player (Figure 3.1 component 5) and a HTML Player (Figure 3.1 - component 6). The media players are responsible for playing their medias and for triggering events on transitions.

In addition to the media players, there is a Context Player (Figure 3.1 - component 7) that is located in the Control Layer (Figure 3.1 middle). The Context Player is not related to any HTML5 media - it represents an NCL context node. It contains references to the players of its nodes – Media or Context. The context player is placed on this layer because it is not directly related to "visible" elements in the presentation — it has a controller role.

The Control Layer also contains the Event Manager (Figure 3.1 - component 8), Interaction Manager (Figure 3.1 - component 9), Player Manager (Figure 3.1 - component 10) and Synchronization Manager (Figure 3.1 - component 11).

The Event Manager is the core of the presentation machine and implements the condition-action mechanism of NCL links. The Interaction Manager is responsible for handling the user's interaction via keyboard, virtual remote control, mouse and touchscreen devices. The Player Manager controls the creation and destruction of players objects and keeps track of all active media players. Finally, the Synchronization Manager is responsible for keeping the presentation synchronized when delays or errors occur due to network problems.

The remaining of this section explains how some important mechanisms work in the WebNCL implementation.

3.4.1 Initializing a Presentation

When an NCL document is loaded into the WebNCL, it is transformed into JavaScript objects from the *NCL Representation Model* component by the *NCL Parser*. Once the document has been translated, the *Player Manager* creates a *Context Player* for the top NCL document context node (<body>). It also creates players for all child nodes (medias and other contexts) of the NCL top context node. The media players start preloading the media, but the nested context players are not completely initialized – their internal nodes are only initialized when the context receives a play event. Each media player is associated with a HTML5 <div> element that contains a media representation tag (<img>, <audio>, <video>, etc.). The initial values properties and other specific properties of the media assume the values of its associated descriptor.

After the body's nodes have been initialized, the *Event Manager* starts to create the event listeners (for transition and user interaction) for the conditions of the body's links. It also creates a function for each link and register it to the corresponding event listeners.

The presentation starts only when the *Synchronization Manager* notifies that all media players are ready. When this takes place, the Event Manager triggers start events over the medias that are referred by the body's ports.

3.4.2 Link Processing

Listing 3.1 depicts an NCL link element named *sampleLink* and the connector *onBeginStart*; Figure 3.2 illustrates the processing of this link by the presentation machine using a simplified sequence diagram.

Listing 3.1: NCL Link

```
<causalConnector id="onBeginStart">
        <simpleCondition role="onBegin"/>
        <simpleAction role="start" max="unbounded"
                qualifier="par"/>
</causalConnector>

<link id="sampleLink" xconnector ="onBeginStart" />
        <bind component="video1" interface="area1"
                role="onBegin" />
        <bind component="image1" role="start" />
</link>
```

When the Video Player of the media named *video1* reaches the initial point of the anchor *area1*, it triggers an event (Figure 3.2 - step 1). Since there is a link that uses this transition as a condition (Listing 3.1, lines 7-8), the Event Manager creates a listener to that event beforehand. The listener executes the registered link functions, for instance, by calling the function *sampleLink* (Figure 3.2 - step 2).

The *sampleLink* function checks if the link's conditions have been met. When the condition are met, the link functions trigger events to their actions. In this example, *sampleLink* has just one condition and it was met, so the *sampleLink* function will trigger events for its actions (Figure 3.2 - step 3). *Image1* Player receives the event and processes it, making *image1* show up.

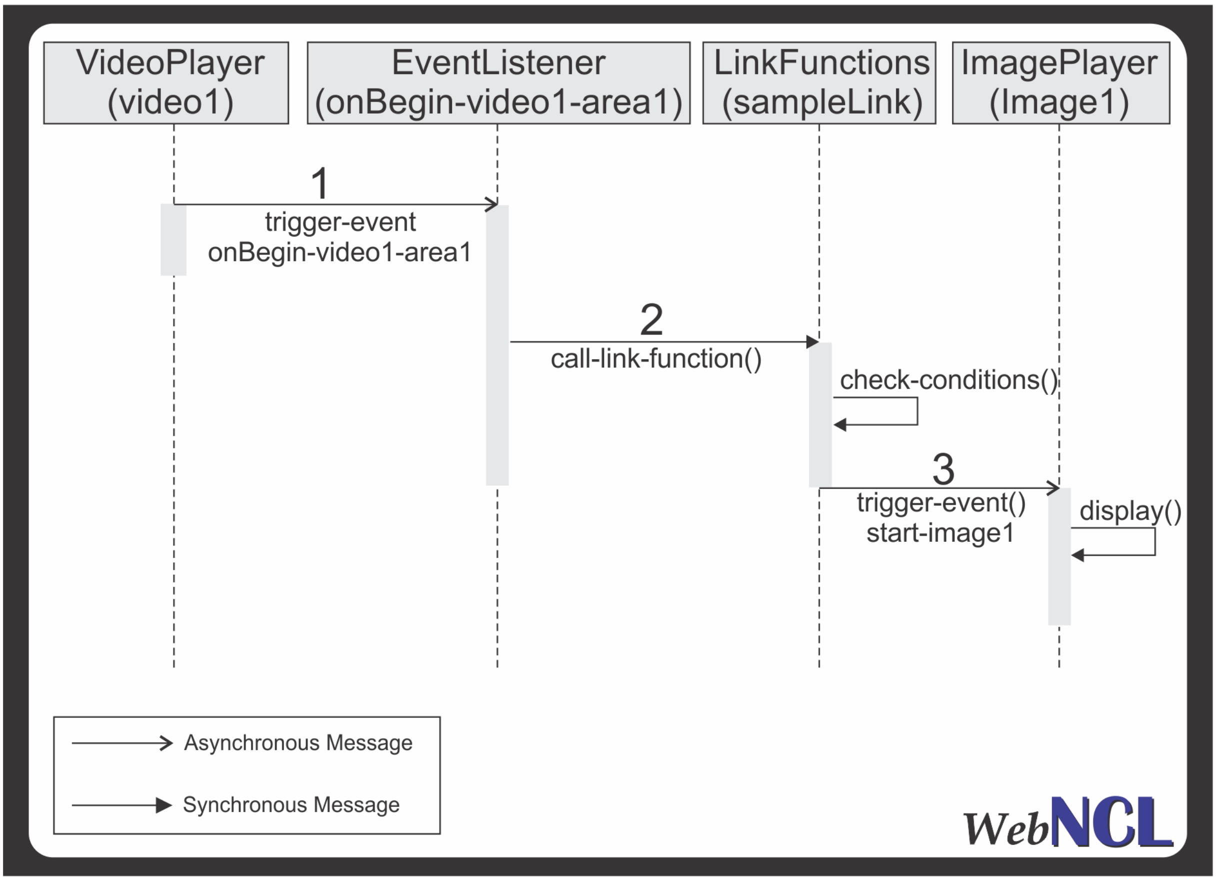


Figura 3.2: Link Processing Sequence

This process is more complex when the link has a compound condition. For compound conditions that uses the AND operator the link function keeps track of the conditions met using a condition table. It registers the time when the first condition is met. When the link function is called by the event listener, it verifies the registered time — if it took place more than one second earlier, the function clears the condition table, registers the current time and checks the table for the condition entrance — in other words, conditions which are too old are discarded. If the time stored is not older than one second, the function checks the condition table for that condition. When the condition table is fully checked, the link function clears the condition table and processes the link actions.

For compound conditions that use the OR operator, the link function registers the last time when the actions were performed. When an event listener calls the link function, it verifies if the time registered took place more than one second earlier; if so, the link function processes the link actions and stores the current time. If not, the link functions just returns —- in other words, the link trigger is ignored because it has just been executed.

The presentation machine also supports links that construct complex conditions by cascading different types of compound conditions.

For actions, the presentation machine supports both PAR and SEQ operators. For actions that use the PAR operator, the link function triggers the events and lets the browser scheduler handles the actions as parallel as it can. For actions that use the SEQ operator, only the first event is triggered, but this event carries a callback function. When the event is processed by Media Players, the callback function is called. This callback function triggers the next event, and this event has a callback function that will trigger the next event, and so on. Using those callback functions the presentation machine is capable of processing link actions in a sequential way.

We adopted a compound condition window of one second to trigger the actions because triggering condition events are not atomic operations. The events may be triggered in a very short window time, making us consider these events as simultaneous. Similar strategy is found in others implementations, such as the Ginga-NCL reference implementation (SOARES; RODRIGUES; MORENO, 2007).

User's interactions are implemented as they were transition events. When the user interacts with the presentation, an event is triggered. If there is a link that has user interactions as its conditions, the event will be handled by an event listener.

3.4.3 Synchronization Mechanisms

To keep media synchronization, even under network fluctuations, a sync mechanism was developed to pause the presentation when one or more players face some problem and resume it when all players are ready again.

The mechanism is first activated when a context starts – including the body, as this is the moment when the players start fetching the media to play. The presentation is paused and waits until all players have their media ready. The mechanism will also be activated when any player — usually continuous media players receiving their media by streaming —- triggers an event notifying a problem to play its media.

The *Synchronization Manager* receives the event and forces a pause in the presentation. When the player triggers another event notifying that it is ready again, the *Synchronization Manager* resumes the presentation.

3.5 Embedding WebNCL in a webpage

In order to present an NCL Document using WebNCL, it is necessary to embed it on a web page, as seen on the HTML page depicted in Listing 3.2.

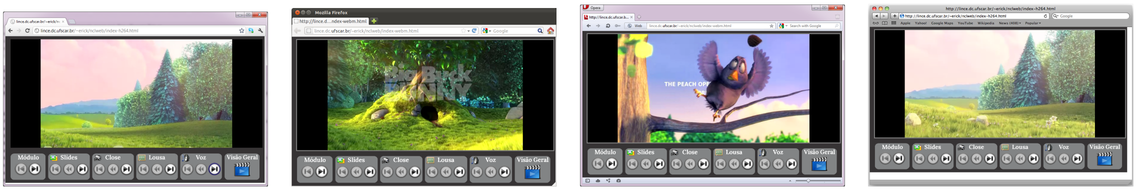
Listing 3.2: HTML code to embed the WebNCL

```
<html>
        <head>
                <script type="text/JavaScript"
                        src="js/WebNCL.js">
                </script>
                <style type="text/css">
                        .nclPlayer {
                                position: fixed;
                                left: 0px;
                                top: 0px;
                                width: 854px;
                                height: 480px;
                        }
                </style>
        </head>
        <body>
                <div id="ncl" class="nclPlayer"/>
                <script type="text/JavaScript">
                        var p = new WebNclPlayer("main.ncl",
                                "ncl");
                        p.start();
                </script>
        </body>
</html>
```

In Lines 3-5 in Listing 3.2, the WebNCL source code to be imported. Between lines 6 and 14 the settings of appearance and positioning are defined as CSS style. Line 18 instantiates the presentation machine, passing the path of the NCL Document to be presented and the id of the HTML <div> where the presentation will be displayed. Finally, in line 21, the presentation is started. More than one WebNCL presentation machine can be embedded in the same HTML page.

During the experiments, an NCL Document with about 4,800 lines and 173 media nodes (videos and images) ran smoothly when submitted to WebNCL. The WebNCL was tested in several browsers and platforms. Figure 3.3 depicts the WebNCL running in different browsers in PC enviroment: Chrome in Windows (Figure 3.3(a)), Firefox in Linux (Figure 3.3(b)), Safari in Mac OSX (Figure 3.3(d)) and Opera in Windows (Figure 3.3(c)). Figure 3.4 depicts the

WebNCL running in an Android device and Figure 3.5 ilustrates the execution in an iOS device. Any combination of browsers and platforms is possible.



(a) Chrome (b) Firefox (c) Opera (d) Safari

Figura 3.3: WebNCL running on PC browsers

3.5.1 User Interaction

Given that many NCL applications use a TV remote control as its primary input device, we have to provide a similar way to interact with these applications. This can be done using different approaches like mapping the TV remote control on keyboard and via a virtual remote control. We implemented both solutions in the WebNCL.

Table 3.1 describes the mapping of TV remote control on a standard PC keyboard. When a mapped key is pressed, the Interaction Manager triggers the corresponding event.

TV Remote Control Button	Keyboard Key
Numbers 0-9	Numbers 0-9
Arrows	Arrows
Enter | OK	Enter
RED	Q
GREEN	W
YELLOW	E
BLUE	R

Tabela 3.1: Keys mapping

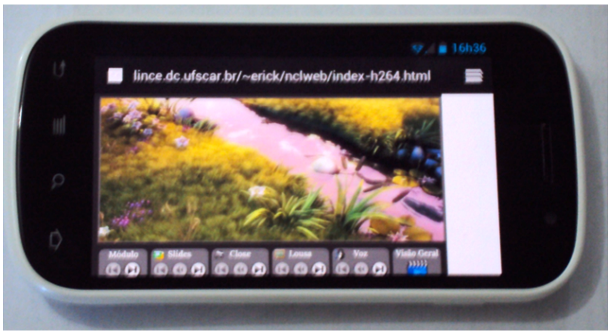


Figura 3.4: WebNCL running on Android

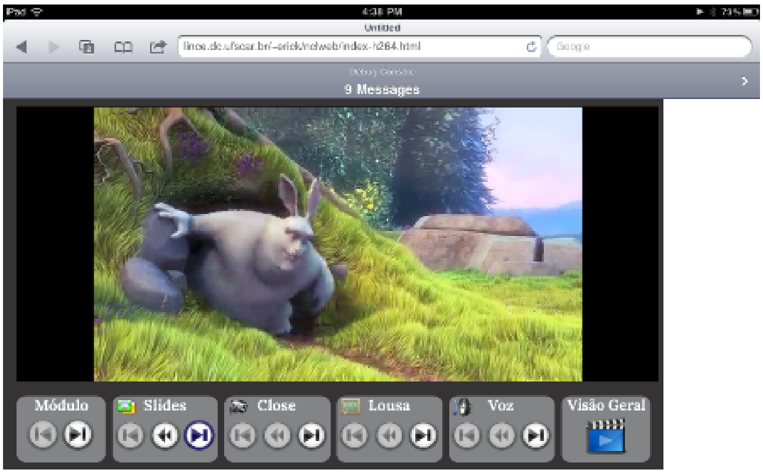
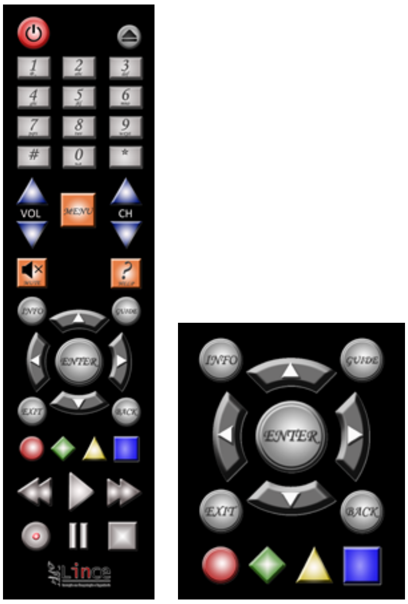


Figura 3.5: WebNCL running on iPad



(a) Complex (b) Simple

Figura 3.6: Virtual remote controls

The virtual remote control is built externally to the presentation machine using HTML and JavaScript. It uses the presentation machine JavaScript API to post key events. This decoupling allows creating different virtual TV remote controls without modifying the presentation machine. For instance, Figure 3.6(a) presents a more complete remote control while 3.6(b) presents a simple.

We also added support to interaction via mouse and touchscreen devices, since the WebNCL can run in environments with richer input devices. When the mouse is over a media, that media receives the focus. If the left or right button is pressed, an OK button event is triggered. In touchscreen devices, tapping in a media without the focus is equivalent to change the focus to that media. A second tap (or double tap) on that media triggers an event equivalent to pressing the OK button.

Although we used rich input devices just to mimic interactions which are already possible with TV remote controls, it is possible to extend the interaction paradigm to support new types of interaction which are more natural in PC or tablet environment (long tap, swipe, double click, drag and drop). In work in progress we are adding support to these interactions in WebNCL.

3.6 Discussions

In this paper we present a solution for building a declarative presentation machine based on web-technologies. Although the WebNCL can allow the execution of NCL documents in any platform that offers a HTML5-compatible browser, it also can also be used as a component of embedded DTV middleware, since the web stacks are compatible with many embedded platforms.

However, the execution of HTML based applications is usually done in a sandbox environment. The application does not have access to the same lower-level APIs available to native codes. Efforts have been made to provide some new APIs to web-browsers, allowing the applications to access accelerometers, GPS, camera and local storage. It accomplishes many requirements of some applications but for some we still need to access native resources. This is the case of the DTV environment – there is a common core that is very platform dependent and that provides very specific services required by DTV applications (for instance: access to the EPG data).

In order to use WebNCL as a component of an iDTV middlware, we must provide low level service access to the presentation machine. Some approaches may be considered in order to extend the WebNCL to provide specific DTV services:

- The HTML5 engine may be embedded in native software, using a HTML rendering engine. This engine can be extended to provide new APIs that bridges the HTML application to the native code. This strategy has been adopted in the mobile environment with the PhoneGap framework (CHARLAND; LEROUX, 2011), which provides some APIs to handle mobile common features, allowing the same code base to run across many platforms (iOS, Android, Symbian, WebOS, Windows Phone).

- Some web-browsers provide ways to create native extensions. The most common way is using the Pepper Plugin API[6] which is a cross platform API to build native client web browser plugins. It allows the user to execute native compiled code in the browser, which runs in a sandbox environment that tries to avoid malicious codes from damaging the system.

- Existing plugins available may be used, such as Java Applets. Specifically in the DTV environment this strategy can be used to integrate declarative applications with procedural Java applications based on GEM Framework (ETSI,) or Java DTV API (JAVADTV,

[6] https://developers.google.com/native-client/overview

2010). Java applets allow the execution of Java applications that are able to access protected resources (out of the browser sandbox).

3.6.1 HTML5 browser issues

WebNCL relies on the web-technologies stack. The browser imposes restrictions regarding the supported media types and its behavior. Currently, there are some limitations imposed by the browsers:

- The iOS devices are limited to playback a single audio or video stream at any time. Playback more than one video at same time is not allowed. The pre-fetching of streamings is also not supported by that platform.
- The video in HTML5 is not supported in the same way by different browsers. Only some browsers support the H.264 encoder, while others support the WebM and OGG encoders. The same goes for <audio> tag.
- The CSS3 properties are handled in different ways by the browsers. For instance, it may impact on how NCL transitions will be presented.
- Browsers eventually implement the Same Origin Policy[7], which avoid a web application to access local files. In some cases, to run local NCL applications it is necessary to access the WebNCL from a Hypertext Transfer Protocol (HTTP) Web Server or disable that browser security mechanism.

3.7 Final Remarks

This paper discusses the possibility of increasing the portability of declarative synchronization languages presentation machines by using web-based implementations. As a proof-of-concept we developed the WebNCL, an NCL presentation machine that presents NCL documents in HTML5 compatible browsers.

The WebNCL brings some implications and possibilities. TV stations may allow users to access the same interactive content broadcasted on TV in the web environment. The same interactive experience can be offered to web users. For instance, a soap-opera can use the same application for viewing characters profiles in both platforms. Another possibility is to offer the same interactive merchandising application in iDTV and Web.

[7] http://www.w3.org/Security/wiki/Same_Origin_Policy

The development of a toolkit based on WebNCL for NCL document authoring is interesting, since, according to the authors experience, the difficulties to setup an devlopment environment may be reduced to developers when compared to traditional virtual machines and emulators.

Given the possibility to embed the NCL presentation machine into webpages, many applications can be conceived. For instance, the Club NCL[8] can provide features to online previewing NCL documents. Educational tools could incorporate interactive videos in a lightweight approach using WebNCL. The traditional video lectures may be replaced by interactive multimedia lectures. This approach is being exploited by the authors.

WebNCL also helps the exploration of the multiple devices NCL support. As WebNCL runs on any HTML5 compatible browsers (which are available in most modern mobile devices) the efforts to integrate the devices — set-top-boxes and mobiles — can be smooth.

Currently, we are extending WebNCL to provide support to the Lua scripting language and implementing Lua APIs specified in ISBD-T. Through these efforts we hope that applications developed for the Ginga-NCL environment can be entirely reused in the web environment.

WebNCL is open-source and the source code and demo applications are available at http://webncl.org.

[8] Club NCL is a portal to share NCL applications - http://club.ncl.org.br/

Capítulo 4

Go beyond boundaries of iTV applications

The development of multimedia applications that require the manipulation and the synchronization of multiple media and the handling of different types of user interactions usually requires specialized knowledge in imperative languages. Declarative languages have been proposed in order to make this task easier, especially when applications are restricted to certain classes, as it is the case of Interactive TV applications in which user interactions are restricted to a few simple models. However, those simple models may be too simple when documents are reused in other platforms: for instance, when watching a video most web users expect an interactive timeline to be available — which is not the case in interactive TV videos. This paper presents a component-based approach to the enrichment of declarative languages for multimedia so that desirable user-media interactions are made possible at the same time that the original ease of authoring is maintained. We detail the components and present a corresponding proof-of-concept prototype. We also discuss design decisions associated with the development of the components, which should be useful in further extensions.

4.1 Introduction

The design of multimedia presentations or applications — for the web, desktop, mobile devices or interactive TV (iTV) — is an interdisciplinary activity which demands professionals from different fields such as communication, design, marketing and art. The development of multimedia applications, however, requires the work of professionals with good programming skills, usually in imperative languages.

Therefore, good communication and understanding among computer professionals and others involved in the creation of multimedia presentations is essential to produce applications faithful to the wishes of their authors.

The possibility of the same professionals that design multimedia presentations be also able to undertake part of its implementation, even in draft or prototype versions, can be productive, for instance to improve the communication among programmers in terms of improvements. Moreover, if the final implementation can be carried out by the creators of the presentation, economic gains can also be obtained. Furthermore, the easy authoring of multimedia presentations may result in an increase in the amount of presentations created.

Declarative languages for the specification of multimedia applications such as SMIL (BULTERMAN; RUTLEDGE, 2008), for web applications, and NCL[1] (ABNT, 2007; SOARES; MORENO; NETO, 2010), for iTV applications, were proposed to facilitate the authoring of multimedia applications — when compared with authoring with imperative languages. However, given that these languages provide good options for synchronizing media but fewer options for user interaction, a limited class of multimedia applications is supported — at least as far as the ease of authoring is concerned. It is not trivial, for example, to create declarative applications that allow media annotations or, in the case of NCL, use a time-slider to allow usert to interact with continuous media — even though the latter is an operation which many Web users are used to.

The opportunity for authoring interactive multimedia documents automatically by combining information captured from live experiences has been extensively exploited — in particular considering the lecture environment. In this case, information captured in a classroom (video, audio, slides, electronic ink, etc.) is used in the composition of interactive web-based multimedia documents (ABOWD et al., 1996; HüRST, 2003; LANIR; BOOTH; TANG, 2008; LIENHARD; LAUER, 2002).

Several platforms exist that allow, more or less automatically, combining captured information into web-based videos so that the result is a multimedia application usually encoded in HTML5 or Flash. Examples in the educational domain include Coursera[2], EDx[3] and EyA (**??**). However, real world demands require that the generated multimedia documents be made available not only for the web and mobile platforms, in which HTML5 would suffice, but also for the iTV environments. This lead us to investigate how to automatically generate interactive multimedia applications encoded in the NCL, since NCL interactive multimedia documents can be used both in compliant native iTV environments and in the web (using the WebNCL

[1] Declarative language adopted by the ISDB-Tb (International System for Digital Broadcast, Terrestrial, Brazilian version) and by the ITU-T for IPTV.

[2] www.coursera.org

[3] www.edx.org

engine (MELO et al., 2012)).

A challenging requirement, given the many platforms available, is the production of multimedia documents that offer interaction facilities expected by users of all platforms — web users, for instance, are likely to be frustrated when facing a video without a time slider. Such requirement challenged us to build components that enable the necessary interactions at the same time that maintain the original declarative authoring approach. In this paper we present a component-based approach to the enrichment of declarative languages for multimedia so that desirable user-media interactions are made possible at the same time that the original ease of authoring is maintained. We also present a corresponding proof-of-concept prototype and discuss design decisions should be useful in future extensions.

This paper is organized as follows: in the next section we discuss related work, especially research related to simplicity in authoring of multimedia applications; in Section 4.3 we discuss both some interesting features for multimedia applications which have proven to be difficult to be implemented with declarative languages, and how these difficulties can be overcome; in Section 4.4 we present components that generate NCL + Lua code that implements these features; in Section 4.5 we illustrate the use of the components by means of the generation of a highly-interactive NCL Document. In Section 4.6 we present our final remarks.

4.2 Related Work

Declarative languages for authoring hypermedia documents present some limitations that can usually be overcome with aid of imperative script languages. Efforts for reducing the dependence of script languages are reported by King et al. (KING; SCHMITZ; THOMPSON, 2004): the authors add to SMIL some capabilities of functional languages, such as expression evaluation and value-based reactive events. Using a similar approach, Soares et al. (SOARES et al., 2010) add to NCL the capability for store and retrieve values in variables. The authors also extended the NCL's causality relations to support variable-based conditions. Both works reduce the necessity for imperative code, but they achieve this by incorporating in declarative languages low-level features of non-declarative languages.

The development of modern multimedia application for the Web can be carried out using HTML5, CSS and JavaScript. Although the content, structure and style of applications can be defined using declarative languages, time (and complex spatial) synchronization must be done with aid of JavaScript. The work of Cazenave et al. (CAZENAVE; QUINT; ROISIN, 2011b) combine the flexibility of HTML5 and CSS with the SMIL Timing and Synchronization module in order to avoid using JavaScript code to perform synchronization in Web applications. The work represents an enhancement in the development of HTML-based applications, but does not offer an easier way for developing multimedia applications than the one offered by SMIL.

The work of Azevedo and Soares (AZEVEDO; SOARES, 2012) extends NCL in order to add support for 3D scenes described in the declarative language X3D. The authors propose the use of X3D documents as NCL media nodes, mapping NCL anchors into equivalent elements of X3D documents. They also extended the NCL's causality relations to add some conditions related to 3D scenes, such as collisions. Although the combination of X3D and NCL allows the creation of complex and rich multimedia applications, composed by image, videos and 3D scene, it does not enhance the ease of authoring of multimedia applications.

One issue present in some declarative languages, especially in NCL, is the verbosity and recurrence of constructions. This lead to template-based generators such as XTemplate (SANTOS; MUCHALUAT-SAADE, 2012) and Template Authoring Language (TAL) (NETO; PINTO; SOARES, 2012). These works specify languages for writing templates that can be imported into documents for further generation of multimedia applications in languages such as NCL or HTML. Although these works enhance the expressivity of the declarative languages by offering templates as high-level construction blocks, they lack in providing high-level interaction models such as timeline navigation.

Another common approach to ease the development of multimedia applications is the use of graphical tools such as SMIL builder (BOUYAKOUB; BELKHIR, 2011), EDITEC (DAMASCENO; SANTOS; MUCHALUAT-SAADE, 2011) or FIND (RODRIGUES et al., 2012). SMIL builder creates and validates SMIL documents using a visual representation of the SMIL timeline expressed in a specialized Petri Net. EDITEC allows the authoring of NCL documents using predefined constructions which are written in the XTemplate language. FIND proposes that authors add complementary content to video by means of annotations, and was inspired in the watch-and-comment approach (**??**) proposed to allow viewers to add annotations at the time of playback. FIND offers clues about moments in which annotations can be added, such as silence moments. Overall, these tools speed up the development of multimdia applications, but they may lead to a limited use of the expressivity of the declarative language.

Meixner and Kosch (MEIXNER; KOSCH, 2012) propose a specific XML language to represent interactive non-linear video. The language maps the video reproduction into a flowchart at the same time that supports the synchronization of additional media with parts of the video. The SIVA Suite (MEIXNER et al., 2010) is the authoring tool and player for non-linear video resulting provided by the authors, who also provide a player for android smartphones (MEIX-

NER; KöSTLER; KOSCH, 2011).

Tondorf et al. (TONNDORF et al., 2012) propose a system for authoring and delivering multimedia problem-solving content. The authors noted the lack of some features in declarative language, such as indexed search for content, and implemented a specialized player with such features which is associated with a XML-based declarative language.

The work of Meixner and Kosch (MEIXNER; KOSCH, 2012) and Tondorf et al. (TONNDORF et al., 2012) are examples of results that ease the development of declarative language-based complex multimedia applications using for specific classes of applications.

The use of layered frameworks such as Django[4] or Grails[5] for speeding up the development of web applications is a common practice. Some researchers applied the same concept for the development of multimedia applications (e.g. (BOLL, 2003) and (LIN et al., 2012)). However, while the first is more concerned with content adaptation among different devices than with easing the authoring task, the latter is based on an oriented-object approach rather than declarative languages.

4.3 Challenges in the Development of Declarative Presentations

For interactive multimedia presentations specified in declarative languages such as NCL and SMIL, corresponding presentation machines machines exist which run on top of standard web technologies (GAGGI; DANESE, 2011; MELO et al., 2012). As a result, the presentations can be deployed in any platform with a compatible browser including tablets, smartphone and Smart TVs.

However, declarative languages for hypermedia synchronization are usually designed to meet some common requirements and constrains of the platforms they are originally intended for. For instance, NCL 3.0 is design for iDTV and IPTV systems and the limitations imposed by these platforms may impact in the development of applications that might be reused in other platforms. Moreover, some applications that stand beyond the boundaries of the iDTV systems may be arduous to be implemented in NCL. One example of limitation is that in NCL the user's interaction is based on TV remote control, which is not appropriate for highly interactive applications (PEDROSA et al., 2011).

[4] https://www.djangoproject.com/

[5] http://grails.org/

In our experience with declarative languages for media synchronization (PIMENTEL et al., 2010; **??**; MELO et al., 2012), we identified some features common to multimedia presentations that are not trivial to be implemented in NCL because of its limitations. Other features are easy to be implemented, but demand a great effort in defining language entities (SANTOS; MUCHALUAT-SAADE, 2012).

By exploring NCL's expressivity, its integration with imperative Lua scripts and a bit of creativity, we were able to implement these non-trivial features. However, the complexity of the implementations is far from the expected for a declarative language. Next we detail the features we identified as hard or verbose to be implemented in NCL, along with the corresponding implementation solutions we offer.

4.3.1 Time-related Navigation

This functionality is common in multimedia presentations generated from the capture of human activities, such as recorded lectures or video conferences ((VEGA-OLIVEROS; MARTINS; PIMENTEL, 2010), (DICKSON et al., 2012)). The presentations usually contain events such as *slide transition* or a *keyword spoken* and the users can navigate to the moment in which these events occur. For instance, one may navigate until the beginning of the previous slide transition or just after the next keyword.

The main difficulty in implementing this functionality using a declarative language lies in the necessity of keep tracking of which are the next and previous events for the current time in the presentation. This requires a complex control in which a variable registers what the current event is as the time advances during playback or the user interacts with the presentation. Since we are working with declarative languages, it is also necessary to consider all the possible values that the tracking variable can assume in order to perform the correct navigation for the next or previous event.

Time-related navigation can be implemented in NCL by mapping each of those events to temporal anchors in a media node. If it is necessary to synchronize different content by these events — such as different video camera streams that frame a classroom — all the related media nodes should also have temporal anchors.

In order to keep track of the events that correspond to the current time, the NCL variable capabilities can be used (SOARES et al., 2010). A virtual media node is (declared and) used to store the variable which tracks the current state. Moreover, a link for each temporal anchor is also created to set the value of the tracking variable when the anchor begins (e.g. the associated event begins).

The actions that users can perform, such as *advance to the next slide transition*, can then be mapped into a button in the application's interface. When the user clicks or taps on this button, a NCL link, similar to the one presented in Listing 4.1, is triggered for each possible value that the tracking variable can assume. Upon checking the value of the tracking variable, only one of the links has its conditions met: this link then starts the media in the corresponding temporal anchor.

Listing 4.1: A "next slide" link

```
<link xconnector="...">
    <bind role="onSelection"
            component="btnNext"/>
    <bind role="propertyType"
            component="event_index"
            interface="last_slide">
        <bindParam name="testValue"
                value="slide_1"/>
    </bind>
    <bind role="abort" component="video"/>
    <bind role="start" component="video"
            interface="slide_2"/>
</link>
```

4.3.1.1 Multiple Navigation Indexes

Applications may allow users to navigate through different types of events. A user may opt at one time to navigate using type (or set) of events (e.g. *slide transition*), in another time a different set of events may be used (e.g. *annotations*), an in another time the union or intersection of the sets may be possible. In this work we refer to each set of events as a *Navigation Index*.

The control of navigation using different indexes is a complex task to be specified using a declarative language. Many possible situations should be considered, including the selection of the index an user may choose for navigation. This can be solved by mimicking a radio button. An image can represent the index for navigation. When clicking on the image, a variable is set to store the current active index. Moreover, the navigation links, such as the depicted in Listing 4.1, also need to check the active index variable.

4.3.2 Timeline Navigation

Timelines are common in applications presenting continuous media, as it is the case with YouTube and QuickTime. As a result, users expect to have a timeline in applications which manipulate video. In fact, when more than one video is availabe, users would also expect that a timeline synchronize the multiple continuous media streams as a single synchronous stream. For instance, it would be important to be able to control, with the same timeline, two video streams taken from different cameras that recorded the same scene but from different views.

The implementation of a timeline in NCL, a language that does not have native support for it, is complicated since: (i) some presentation machines do not allow to set the current time of a continuous media directly; (ii) representing the timeline interface with a composition of media would require a complex control; (iii) users are used to interact with the timeline using a mouse or touchscreen in Web and mobile environment. Such facilities are not common in interactive TV environments.

In order to solve (i), we opted to implement a discrete timeline, the same approach used by Vega-Oliveros, Martins and Pimentel (VEGA-OLIVEROS; MARTINS; PIMENTEL, 2010). Instead of allowing access to each instant of the continuous media, the media is split into several short segments (e.g. 1, 5, 10 or 20 seconds). A temporal anchor for each of these segments is added to the media node.

To solve (ii) we used a Lua script to draw the timeline (NCLua Canvas API). The Lua script must know the current time of the media, which can be implemented by using the events.time() function to count time. When the playback of continuous media starts, the Lua script starts to count time. If the playback of the continuous media is paused, the Lua script should stop counting time until the playback is resumed. The Lua script also has a property anchor named *current_time* which must be set when navigating in the media, in order to update the time information kept by the script. For instance, if the user navigates to the second 85, the *current_time* property is set to 85: the Lua scripts then starts counting time from the second 85.

Regarding (iii), we observed that the basic interaction with the timeline can be implemented using buttons such as *next* and *previous*, similar to the used for Time-related Navigation. Each segment of the discrete timeline can be handled as an event from the timeline index. The problem of this approach is that every time the next button is pressed, a number of links equal to the number of segments of the timeline would be triggered. For instance, in a presentation lasting 1 hour and split in 1-second segments, each time the *next* button is pressed, 3600 links would be triggered and checked for the value of the tracking variable so that one single link would be activated.

We avoided checking all possible (thousands of) conditions by using the Lua script as a proxy. The Lua node has “command” anchors for the next and previous buttons and one virtual temporal anchor for each segment. There is a link for each button that performs a start action over the related “command” anchor. Given that the Lua script knows the current time, it can easily compute which is the next/previous segment and, by using the NCLua events API, trigger a start event in its anchor related to that segment. Back to the NCL, there is a link which has the beginning of the Lua virtual temporal anchor as condition to be fired. It does the job of seeking the continuous media (Listing 4.2).

Listing 4.2: Timeline Link

```
<link xconnector="...">
    <bind role="onBegin"
            component="lua_timeline"
            interface="t_segment_0"/>
    <bind role="stop"
            component="video"/>
    <bind role="start"
            component="video"
            interface="v_segment_0"/>
</link>
```

Although this approach allows the user interaction with the timeline, it still does not solve (iii). Since we aim to make the multimedia presentations available in platforms such as the Web or mobile devices, the approach of using buttons to interact with the timeline is unfamiliar to users who are habituated with mouses or touchscreens. WebNCL (MELO et al., 2012) maps mouse and touchscreen interactions into common remote control events. When the mouse is over a media, that media receives the focus. If the left or right button is pressed, an OK button event is triggered. In touchscreen devices, tapping in a media without the focus is equivalent to change the focus to that media. A second tap (or double tap) on that media triggers an event equivalent to pressing the OK button.

In order to offer Web-like mouse interaction with the timeline, we added an invisible virtual media node above each segment of the timeline interface. When the user clicks on these virtual media nodes, a link performs a start action over the related Lua node’s virtual temporal anchor. This action triggers the link of Listing 4.2 in charge of positioning the media in the desired segment.

4.3.2.1 Plotting indexes in the Timeline

It is possible to plot points on the timeline corresponding to events from a time index. Users may navigate to these events by clicking over the related points. In order to implement this feature we added a media for each event over the correct position on the timeline interface. It is also possible to show visual feedback about an event when the user moves the mouse over its media node on the timeline. For instance, in slide transition this visual feedback could be the new slide.

4.3.3 User Query

As an example of user query, an interactive video application that alerts the user about a scene which may contain inappropriate content offers user the option of watching or skipping the scene. As another example, at the end of a music video clip the user may choose which video clip is next.

In fact, asking the user to decide which media to play is a common requirement in many multimedia applications. It is not a complex functionality to implement, but it requires many different links to be implemented: (i) one to show the query, (ii) one to hide the query after a certain time, (iii) one for each possible alternative available.

4.3.4 Spatial Composition Changes

Spatial Composition Change is a common requirement of multi-video applications. For instance, in a camera system several streams may be presented in small windows which share the same screen. It may be possible to maximize one of the videos in order to verify something suspicious, choosing one of the video streams to be presented in a large (main) window while the other videos remain in small windows. Then, the user may select one video of the small windows to be swapped with the video presented in the main window. In a scenario in the educational domain, a multimedia presentation generated from a lecture may be composed of several video streams: one which captured the slides, another which captured the whiteboard or the instructor's computer, and others corresponding the different views of the instructor. The students may choose to focus their attention into one specific stream presented in one large (main) window while the other streams are presented in small windows – and may swap the video presented in the main window with any other. The change in spatial composition is also a requirement for reactive applications that must change their layout in response to user interactions, such as when the orientation of the device changes or the display is resized.

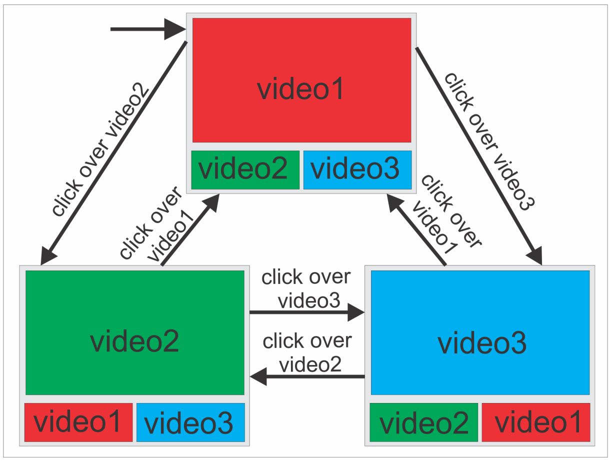


Figura 4.1: FSM for Spatial Composition

The problem on implementing this functionality lies in keeping track the “window” in which each media is presented. Moreover, it may be necessary to declare links for the many possible combinations of media and positions in order to react properly to user’s interactions. We implemented the changes in spatial composition using a finite-state machine (FSM). Each state in the FSM corresponds to a spatial composition the medias can assume. The user’s interactions (or internal events) are the transitions in the FSM.

Figure 4.1 illustrates a FSM for a composition of 3 videos. There are 3 states, each one with one of the videos as the main (larger) video. When a user clicks on a video presented in a small screen, the corresponding a state transition is triggered and that stream is shown in the main video. We implemented this in NCL by using a variable to store the current state. When a interaction corresponding to a transition in the FSM is performed, a link for each transition related to that event is triggered. These links check the current state variable, if the current state has a transition that is triggered by that interaction, the corresponding link performs its actions, changing the video bounds and setting the current state variable to meet the new state configuration.

Note that in the configuration illustrated in Figure 4.1, if the current state is the one with the video2 as the main video and the user clicks over the video2 two links are triggered. However, since the current state does not have transitions related to this event, none of the links perform their actions.

4.3.5 Annotation

User annotation is an important requirement for multimedia applications. In our approach annotations are based on a timeline: the user marks some media segment as important, or adds some comment at a given instant of the media, etc.

One main concern about annotations in multimedia presentations is how to store and retrieve the annotations created by end users. A possible approach is to modify the multimedia presentation on-the-fly, adding new media and links to represent the annotations (CATTELAN et al., 2008; JANSEN; CESAR; BULTERMAN, 2010; PIMENTEL et al., 2010). This could be implemented in NCL by using the Ginga-NCL live editing API (COSTA et al., 2006) and dumping the modified NCL document into a new XML file. A version control mechanism for these annotated NCL files would also be required.

We propose a different approach: a Web Service is used to store and retrieve the annotations. A Lua script accesses the Web Service and, using the data retrieved, draws the annotations over the timeline. In order to add a new annotation, the Lua script must know the current time of the media in which the annotation will be performed. This can be obtained from the Lua script that controls the Timeline Navigation.

The interface between the Lua script and the NCL document is implemented using some virtual anchors, as depicted in Listing 4.3.

Listing 4.3: myListing

```
<media id="timeline" src="timeline.lua">
    <!-- ...  -->
    <area id="important" />
    <area id="doubt" />
    <area id="comment" />

    <area id="next_annotation" />
    <area id="previous_annotation" />

    <area id="annotation1" />
    <area id="annotation2" />
    <!-- ... -->
    <area id="annotation50" />

    <property name="annotation1_bounds"
            value="None" />
    <property name="annotation2_bounds"
            value="None" />
```

```
19     <!-- ... -->
20     <property name="annotation50_bounds"
21              value="None" />
22 </media>
```

We use images as buttons to allow users to create annotations while interacting with the presentation, as suggested by the watch-and-comment paradigm (CATTELAN et al., 2008). For instance, a button may be used to express that the segment is important and other to add that a textual comment. When these buttons are clicked, a link performs a start action over the correspondent virtual anchor of the Lua object (Listing 4.3, lines 3-5).

The Lua script uses the information about the current time to save the annotation in its internal structures (using the discrete segments of the timeline). It also saves the annotation in the web server. The annotations stored in the Lua internal structures are then drawn in the timeline.

It is possible to navigate among the annotations using the next and previous buttons. This is implemented using the Lua script as a proxy (similar to the timeline navigation). The buttons trigger links that perform start actions over the Lua object command anchors (Listing 4.3, lines 7-8) and the Lua script performs a start action over the appropriate Lua virtual temporal anchor.

In order to allow mouse navigation in the annotations, it is necessary to add virtual invisible nodes. Because at the time of the creation of the NCL document we do not know which annotations a continuous media will receive, it is necessary to add some media in advance for this purpose. However, we also do not know which segments will be annotated, which means that the positioning the invisible media in the timeline cannot be made when the NCL document is created.

The positioning of annotation media nodes are computed by a Lua script. The positions are sent to the NCL document when the Lua script performs a set action over one of the annotation bounds anchor properties (Listing 4.3, lines 15-21). The value the Lua script sets in these anchors are the positions in which the associated virtual nodes must be placed. Back to the NCL, a link that is triggered when a value is set to the annotation bounds anchor sets the bounds property of the associated virtual node. When the user clicks on these media nodes, a link performs a start action over corresponding virtual temporal anchors of the Lua object (Listing 4.3, lines 10-13). Finally, the Lua object performs a start action over the proper timeline segment anchor.

When a presentation begins, the Lua script retrieves all the previous annotations made by the user from the Web Service and set their positions in the timeline.

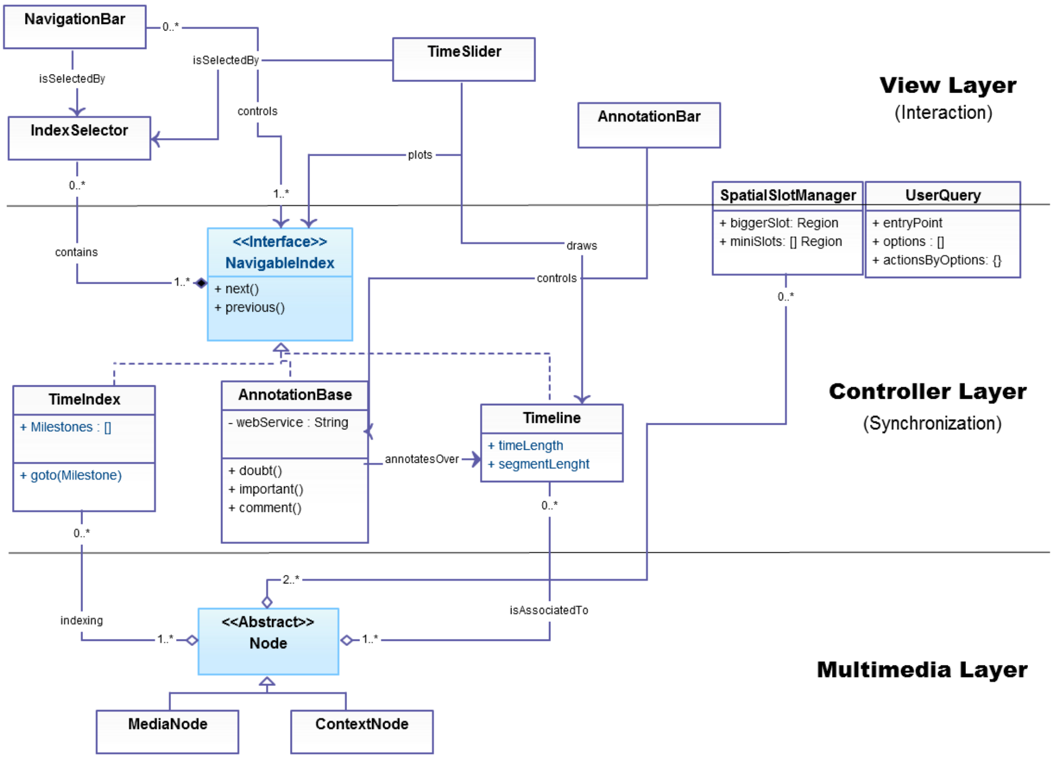


Figura 4.2: Class Diagram of the components

4.4 Components

Despite the fact that the features discussed in the previous section can be found in many classes of multimedia presentations, the corresponding implementations in NCL are non-trivial. In order to ease the authoring, we designed and implemented components in Python that generate NCL + Lua code for these features. Figure 4.2 depicts a simplified UML class diagram of the components.

The components are grouped in a layered architecture inspired in the Model-View-Controller (MVC) design pattern. The bottom layer is the **Multimedia Layer**, wherein the media nodes used in the multimedia application are declared. In the case of a NCL document, they are the media and context nodes. The middle layer is the **Controller layer**, which holds the components that define the types of navigation and synchronization (both temporal and spatial) it is possible to perform with the media nodes. The top layer is the **View Layer**, which contains the components responsible for allowing users to interact with the applications.

In this layered architecture, which we refer to as *Multimedia-View-Controller* (MMVC) architecture, media nodes from the multimedia layer are manipulated by the components of the Controller Layer. The user, via components of the View Layer, may interact with the media. The view layer sends messages to the controller layer which performs the actions to manipulate the media.

Note that in Figure 4.2 the components **SpatialSlotManager** and **UserQuery** are above the line that separates the View and the Controller Layer. This is because the synchronizations defined in these components are intrinsically related to user interactions. For instance, the **SpatialSlotManager** component allows some media nodes to exchange the spatial position among themselves after a user selection. It would be possible to split it in two components the parts responsible for the interaction and responsible to define the behavior, each one in one layer, but these two components would, almost always, be used together.

The components **TimeIndex**, **AnnotataionBase** and **Timeline** implement the interface **NavigableIndex**. These components allow navigation operations such as *"go to the next point"* or *"move forward to the previous point"*.

The **TimeIndex** allows the creation of an array of time milestones for a media or a group of medias. For instance, Listing 4.4 depicts the creation of a time index for a media node. It is possible to bind media nodes via their unique ids in the NCL document (Listing 4.4, line 2) or via an instance of Media class (Listing 4.4, line 3-4). The milestones can be passed as a list of times in seconds or individually. A **NavigationBar** is a visual component that presents buttons like next and previous and allows user navigation by any type of **NavigableIndex** instance. Since it is a visual component, it is necessary to define the position in which the buttons must to be displayed. This is done by passing a NCL region to the component (Listing 4.4, line 7)

By combining the **TimeIndex** and the **NavigationBar** components, it is possible to implement the Time-related Navigation (Section 4.3.1).

Listing 4.4: TimeIndex

```
1 index = TimeIndex()
2 index.bind_media(’video0’)
3 index.bind_media(Media(id=’video1’,
4             src=’video1.mp4’))
5 index.add_milestones([5, 48, 58.5])
6 index.add_milestone(78)
7 bar = NavigationBar(’rNaviBar’)
8 bar.add_index(index)
```

The **Timeline** component bounds a media or group of medias to a single timeline. It also enables the user to navigate in the media using a timeline. In order to do this, is necessary to bound it to a **TimeSlider** component. The **TimeSlider** is a visual component that draws a timeline and can be used to access any instant of a media or group of media. Together, these components implement the Timeline Navigation (Section 4.3.2).

Listing 4.5 depicts the instantiation of a **TimeLine** for a group of media nodes and show how to bind it to a **TimeSlider** visual component. The duration of the **TimeLine** and the length of the each discrete segment (in seconds) are informed in the constructor (Listing 4.5, line 1). It is possible to bind a media node and to define the time instant in which the timeline begins (for instance, the second 0 of the timeline corresponding to the second 80 from the media *video1* (Listing 4.4, line 3). It is necessary to define the NCL region of the **TimeSlider** component, as well the **TimeLine** instance it will draw and the means by which users can interact with the component (Listing 4.5, lines 4-6).

Listing 4.5: Timeline

```
1 my_line = Timeline(duration=600, segment=1)
2 my_line.bind_media('video0')
3 my_line.bind_media('video1', begin=80)
4 slider = TimeSlider(mouse_interaction=True,
5              timeline=my_line,
6              region='rTimeline')
```

Listing 4.6 illustrates the utilization of the components **AnnotationBase** and **Annotaton-Bar**. The **AnnotationBase** allows to create and retrieve annotations made over a **TimeLine**. If, instead of a **TimeLine**, a continuous media is passed in the **AnnotationBase** constructor, a **TimeLine** will be automatically created and bounded with the media (Listing 4.6, lines 2). It is also necessary to pass the URL of the Web Service that will store the annotations (Listing 4.6, lines 3). Since **AnnotationBar** is a visual component, it is necessary to inform the NCL region in which it will be displayed (Listing 4.6, lines 4). These components implement the annotation functionality (Section 4.3.5).

Listing 4.6: Annotation

```
1 note_base = AnnotationBase(
2         timeline='video0',
3         url='localhost:8080/service')
4 note_bar = AnnotationBar(region='rBar',
5         base=note_base)
```

Both the **NavigationBar** and the **TimeSlider** can support more than one **NavigableIndex** instance. The **IndexSelector** component is necessary to allow the user chooses which of the possible index he or she wishes to use for navigation.

Listing 4.7 illustrates the use of a **SpatialSlotManager** component. This component displays a “maximized” media and a group of “minimized” in defined spatial slots. It is possible to maximize any of the minimized medias by clicking on them. The slots (both the bigger and smalls) are defined by a tuple with the slot spatial position (represented by a NCL region) and the media initially bounded to that slot. This component implements common cases of Spatial Composition Changes (Section 4.3.4).

The **UserQuery** can be used to take actions based on options queried to the user (Section 4.3.5). It is necessary to define an anchor that, when activated, will present the component to the user. The options are defined as a list of text or images. After the user choice, among one of the options, the corresponding action is performed.

Listing 4.7: Annotation

```
ssm = SpatialSlotManager()
ssm.bigger_slot = ('rMainSlot', 'video0')
ssm.add_mini_slot( ('rslot1', 'video1') )
ssm.add_mini_slot( ('rslot2', 'video12) )
```

4.5 Validating the Components

We have used the components described in the previous section to build an application that generates NCL documents from captured lecture-style presentations. The result of the capture is a multi-video multimedia containing different views of the lecture, with video streams generated from capturing the slide presentation, the computers used by the instructor, various views of the classroom, etc. There is also an XML file which holds metadata from the captured lecture, such as the time instants in which slide transitions occur, or the lecturer used her computer or the (electronic or traditional) whiteboards. The orchestration of capture process and the metadata extraction is introduced (VIEL et al., 2013c) and detailed (**??**)elsewhere.

Figure 4.3 shows an example of an NCL document generated from the captured presentations. The NCL documents offer facilities that includes the synchronization of the captured video streams. The document in this example has two video streams captured from cameras (Figure 4.3(1) and Figure 4.3(3)), one stream capturing the slide presentation (Figure 4.3(2)) and the instructor’s computer screen (Figure 4.3(4)). They all have the same length and their synchronization is handled by a **Timeline** component.

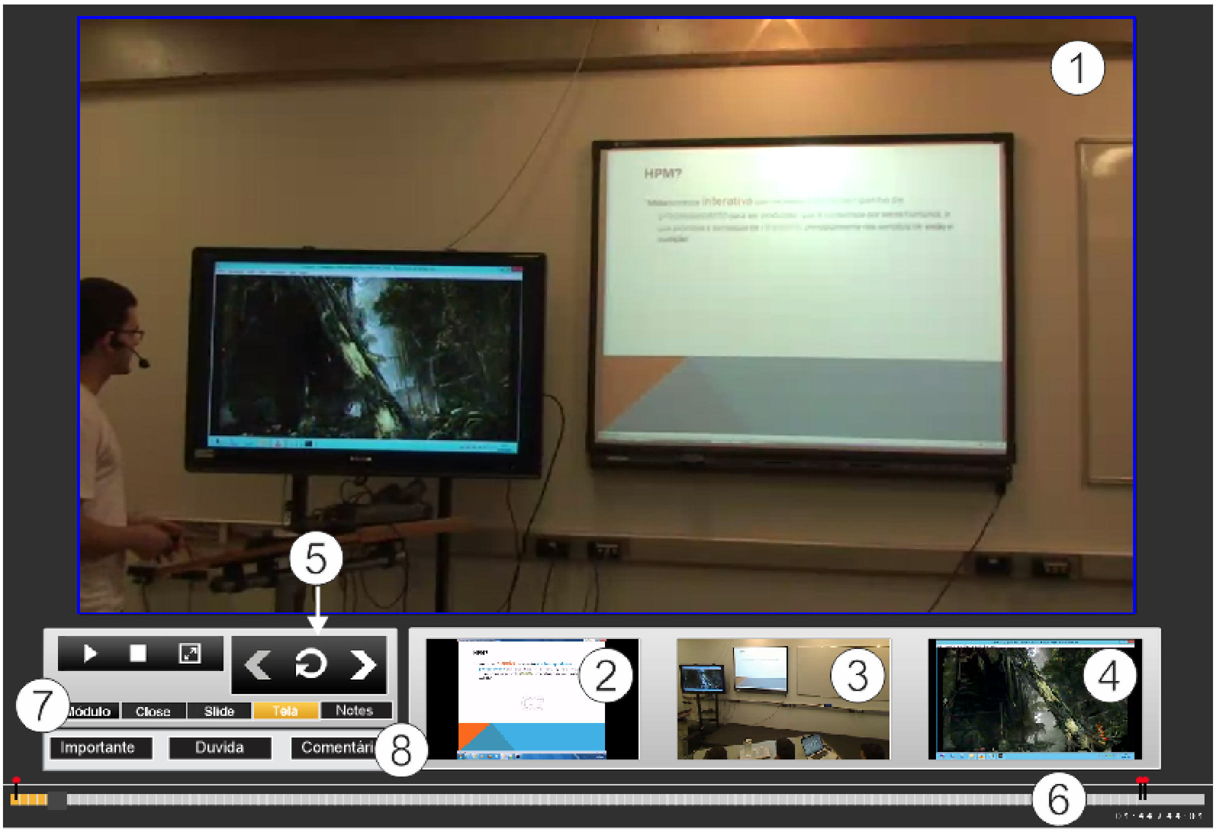


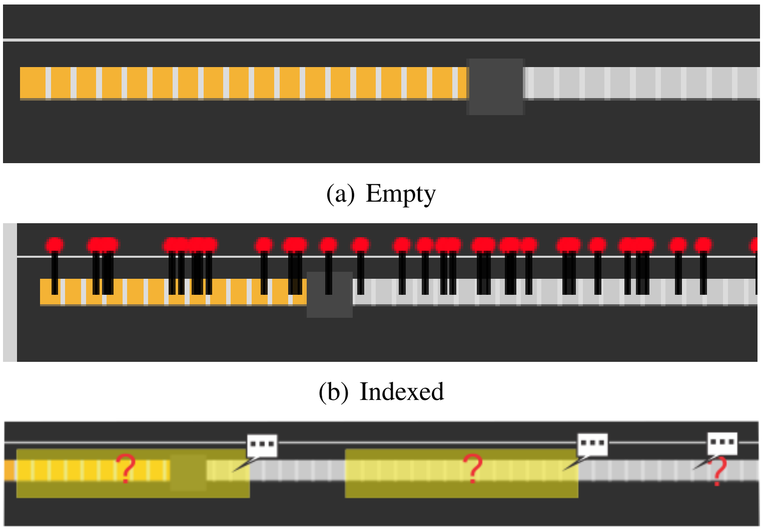
Figura 4.3: Multi-video multimedia object captured from a lecture

Users may interact with the presentation and access any segment of the synchronized video streams via a **TimeSlider** component (Figure 4.3(6)). There are also temporal indexes implemented by instances of **TimeIndex** component. Indexes are provided for slide transition, lecturer's close and computer interactions. User may select an index for navigation using the **IndexSelector** component (Figure 4.3(7)), and the index-based navigation uses the **NavigationBar** component (Figure 4.3(5)).

Users can also annotate the document via the **AnnotationBar** component (Figure 4.3(8)). The annotations are stored in the **AnnotationBase** which saves the information in a WebService.

Similar to co-located lectures, wherein students may focus their attention to different contents (the lecturer, whiteboard, slide presentation, the textbook, etc.), the NCL document allows users to choose which video stream to see in detail in detail. One option is to show the video in the main window using the **SpatialSlotManager** component (Figure 4.3(1-4)); the other option is via the full-screen mode.

The **TimeSlider** component can be used to show annotations and marks or milestones (such as slides transitions) to the timeline interface. If none of the index are selected in the **IndexSe-**



(a) Empty

(b) Indexed

(c) Annotated

Figura 4.4: Timeline

lector component, the timeline will bear no marks as in Figure 4.4(a). When a **TimeIndex** is selected, marks representing the milestones will be added to the timeline (Figure 4.4(b)). When an annotation index is selected, the annotations will be displayed in the timeline (Figure 4.4(c)).

4.5.1 Analysing the NCL Document

To illustrate the complexity of the NCL documents generated with the aid of the components, Table 4.1 summarizes the size the document generated automatically from a captured lecture lasting 1 hour and 18 minutes.

The NCL document uses 48 video clips: four of them can be played back simultaneously. Besides the videos, the document include 1197 other media items, mostly images. There are also 1898 links responsible for controlling all navigation alternatives (including timeline and index-based).

The NCL document itself has about 3.6 MB of size, not considering the videos, images and other media items. The total size of the multimedia document, considering all media, is about 1GB [6]. All the media, anchors, links and the static elements of the NCL (regions, descriptors) add up to a total of 65148 lines of code.

We use the WebNCL presentation machine to render the interactive NCL Documents gene-

[6] Each video has a version coded in H264 AVC and other coded in VP8, with resolution of 854x480 pixels and 24 frames per second. Note that some of the video streams, such as the on corresponding to the slide presentation, have many static segments which lead to small video size when codecs that remove temporal redundancy such as the H264 and VP8 are used.

Tabela 4.1: NCL Document Numbers

Total Duration	78 minutes
NCL Document Size	3.6 MB
Lines of Code	65148
Links	1898
Media Node	1245
Videos	48

rated from captured lecture. Although the document sizes may suggest problems in its execution, the WebNCL has been able to reproduce the NCL Document smoothly.

Some improvements in the components may reduce the size of the generated NCL documents. The current version of the components generate a human readable code for debugging proposes, but renaming the NCL entities with concise names and removing white spaces can significantly reduce the document size. Other optimizations in the NCL code structure can also help reducing the document size.

4.6 Final Remarks

Declarative languages for media synchronization were conceived to ease the development of multimedia presentations. However, declarative languages for hypermedia synchronization are usually designed to meet common requirements and constrains of the platforms they are originally intended for. The reuse of applications developed in declarative languages in other platforms may be negatively impacted by these limitations.

In this work we elected NCL, a declarative language designed for DTV and IPTV systems, to develop applications that stand beyond the boundaries of such platforms. As result, we identified some features, common in general purpose multimedia applications, that are non-trivial to be implemented in NCL. Even though, we keep the choice to take advantage of several interesting features of such declarative language for multimedia synchronization. Although theoretically possible, implementing such features using only standard NCL resources would imply a complexity far from the expected for a declarative language.

In order to provide these features, keeping the ease of development, we implemented components organized in a Multimedia-View-Controller architecture that generate NCL + Lua code which implements the non-trivial functionality. We illustrate the use of the components generating a complex and highly-interactive multimedia presentation from a presentation generated with information captured from lectures.

Although the components presented here are not a comprehensive listing capable of implement many classes of multimedia applications, they are construction blocks that can be used in many common applications.

As a future work, we plan to use the components presented in this work as the base for a layered framework for authoring complex and platform-independent multimedia applications. It is important to note that the solutions proposed here, although implemented for NCL, are sufficiently generic to be exploited in similar situations found in other declarative languages for multimedia applications.

Capítulo 5

An Approach for Controlling Synchronous Remote Instances of a Multimedia Presentation

Be in touch and sharing experiences with family and friends about TV shows and other multimedia content are becoming a trend. The enjoyment of such shared activities depends on how synchronized the users' contents are. Many works propose architectures and algorithms to provide continuous media synchronization in order to enhance the quality of shared media consumption. However, in some applications, such as interactive multimedia presentations, the content synchronization depends on many factors, some of which can be affected by users' interactions. In these applications, in order to provide a synchronous shared experience among users, the interactions of one user should not affect only her presentation, but the presentation of all users which are engaged with her in the same activity. In this paper we propose an approach for engage users in synchronous shared experience for interactive multimedia presentations. Our approach is based on replicating the interactions of one specific user in all presentation instances, in order to keep the presentations synchronized. This means that a master user, like a pilot, controls not only her local presentation, but the presentations of all users. We present some challenges that need to be overcome in order to implement this approach and, for each challenge, we list some possible solutions and their pros and cons. We also present a proof-of-concept for NCL Documents and illustrate its use with highly-interactive multi-video presentations.

5.1 Introduction

The popularity of social networks, the spread of high-speed broadband connections, and the consolidation of video on the Web and Social TV are collaborating to transform media consumption into a synchronous shared experience(GEERTS et al., 2011; VAISHNAVI et al., 2011). Instead of watching media content alone in their own devices, users wish to engage in this activity with their families, friends or coworkers located elsewhere — and meanwhile they are watching, they are talking about the content, enriching their experience.

A common approach for allowing users share the media consumption experience is connecting two or more users through a text chat or video conferencing while they are watching the same content, such as a soccer match or a political debate (WEISZ et al., 2007). This approach is enough to provide synchronous shared experience when the content is not interactive or its interactivity does not affects content's playback or its meaning.

However, when considering recorded interactive multimedia presentations — composed of several videos, images, audios and texts with temporal and spatial synchronization relationships, such as applications written in media synchronization languages such as SMIL or NCL — users usually do not only watch the content passively, but interact with it. For some classes of multimedia presentations, such as non-linear video, the content playback and therefore users' experience are dependent on user's interactions. In these applications, in order to provide a synchronous shared experience among users, the interactions of one user should not affect only his presentation, but the presentations of all users which are engaged with him in the same activity.

A recorded lecture is an appropriate example of interactive multimedia presentation to be consumed together. In order to better explain some subject, a lecture may be composed of one or more videos (teacher's face, teacher's body, whiteboard, complementary videos, etc), images, texts, animations, etc. The teacher and her students download the presentation into their devices and, at a previously scheduled time, they join in a video-conference. When the teacher is presenting a view of a specific moment of the lecture, in the role of pilot of the presentation, everybody (teacher and student) must see the same in their devices. If the teacher comments, via video-conference, the presented moment of the multimedia lecture, all users will be synchronized in that moment, ready to follow the explanation. If the teacher pauses her video, all students' videos should be paused in order to keep the synchronization with the teacher' presentation. The role of pilot may be passed temporally to a student. With this status, it is the student now who determines what everybody will see. This may be an opportunity for him to play some part of the lecture and ask related questions.

Other possible scenario is a guided virtual visit to an art exhibition. The virtual exhibition is represented by an interactive multimedia presentation, mimicking a real museum, with high-fidelity reproductions of pieces of art such as paintings and sculptures. Although users can explore the presentation by themselves, some might prefer being guided by an art expert, who can be a friend or a museum's employee. Visitors and the guide connect via a video-conference and the tour begins. Visitors' presentations follow the guide's presentation. When the guide is seeing a painting (and explaining via video-conference), all the visitors should be seeing the same painting -– otherwise they would become confused. Although the visitors' presentations follow the guide's one, they could still perform some interactions with the art works, such as rotate a sculpture or zooming in a painting.

The two scenarios described above are examples of applications in which, for synchronous shared experience, the interactions can affect the content's meaning. They also follow a pattern: only the interactions of one user (per time) with the presentation need to be replicated by other users' presentations. In this paper, this approach for live presentations synchronization (of recorded multimedia objects) is called *Remote Pilotage*, since one user controls or "pilots" all presentation instances. As the role of *pilot* may be hold by different users over the time, Remote Pilotage may be applied in a wide range of scenarios, providing synchronous shared experience with interactive multimedia presentations.

We describe the Remote Pilotage approach and discuss some challenges that need to be overcome to implement it. As a proof-of-concept, we extend a Nested Context Language (NCL) presentation machine in order to add the support for Remote Pilotage. We evaluate the proof-of-concept by using the Remote Pilotage approach to control multiple instances of a high-interactive multi-video presentation in different scenarios.

Section 5.2 presents related work. Section 5.3 describes in detail the Remote Pilotage model. Section 5.4 presents some challenges that must be overcome in order to implement the Remote Pilotage and some solutions. Section 5.5 presents the proof-of-concept implemented. In Section 5.6 we present the remarks and future works.

5.2 Related Work

The work of Repplinger et al. (REPPLINGER et al., 2009) adds to X3D support for distributed media playback. It enables the synchronous playback of continuous media into distributed X3D scenarios and allows simple media controls such as play, pause and choosing the media to be played.

In Boronat et al. (BORONAT et al., 2012), the authors propose modifications in RTP/RTCP in order to allow media synchronization in different domains. Thus, two users watching a soccer match from different content providers, such as cable television and a video streamed over a 3G connection, would have their TV show synchronized, despite the difference in latency between content providers.

The works of Vaishnavi et al. (VAISHNAVI et al., 2011) and Geerts et al. (GEERTS et al., 2011) discuss human and technical aspects in order to provide a satisfactory experience on watching video or TV shows together over IP networks.

All the aforementioned works only consider distributed media synchronization for one media stream. They do not address complex multimedia presentation, in which the user interaction is part of the shared experience.

Mauve et al. (MAUVE et al., 2004) and Fleury et al. (FLEURY et al., 2010) present mechanisms to maintain the consistency in distributed applications. Mauve's paper addresses scenarios in which one application state is replicated to all other instances. The work of Fleury is more concerned with collaborative environment, in which all the users can affect the virtual environment simultaneously. Although these works present techniques for synchronous applications, the techniques are not discussed in the context of multimedia presentations.

In Viel et al. (VIEL et al., 2011), the authors present a framework that allows user from interactive Digital TV (iDTV) and mobile platforms interact with remote instances of virtual reality (VR) applications. Techniques to reduce or fix possible inconsistencies between the remote VR application and the users' view are discussed. In Viel et al. (VIEL et al., 2012) that work is extended by allowing the use of remote VR applications as a media in a multimedia presentation. However, these works do not address coherence among different media in a distributed multimedia presentation.

Neto et al. (NETO; SANTOS, 2008) presents an approach in which broadcasters can control, during live TV shows, multimedia presentations in their clients' set-top-boxes (STB). That work can be viewed as an instantiation of the remote pilotage, but it lacks the generality of the approach proposed by this work.

In Boronat et al. (BORONAT; LLORET; GARCíA, 2009) the authors present a survey of techniques for multimedia synchronization, including scenarios in which the users' interactions can affect playback. However, that work does not address declarative synchronization languages.

5.3 Proposed Approach

A *Remote Pilotage Session* can be defined as a group of users interacting with the same multimedia presentation at the same time. Only one of the users, per time, is in the role of pilot of the presentation. The users might be related to each other, like friends in a social network or students of the same course. They might also be strangers with a common interest, such as fans of the same sport team.

Note that in a Remote Pilotage Session, the multimedia presentations watched by the users do not need to be strictly equal. They need only to be equivalent, which means they should be composed by the same (or equivalent) media and offer the same types of interactions in order to be controlled in a similar way. For instance, presentations developed for different platforms, such as iDTV and mobile devices, are different. Their medias are of different resolutions and may be presented in different spatial dispositions, but the presentations are considered equivalent if the content of their medias are similar and if they share the same interactions. Other example is a presentation with higher contrast to users with some visual impairment, which is different but equivalent to a presentation with normal contrast values. Still, different captions or audio streams may be adopted by equivalent presentations.

Users that engage in a Remote Pilotage Session can assume 3 different roles, as detailed in the UML User Case depicted in Figure 5.1.

The *Manager User* is responsible for creating the session (e.g. choose which multimedia presentation will be presented). He has also the responsibility to determine the moment in which the session begins and ends. When session begins, the *Manager* also holds the role of *Pilot*. He can promote one of the *Spectator Users* to a *Pilot user*, which makes him lose the *Pilot* role. However, he can always retake the *Pilot* role, which makes the previous *Pilot user* becomes a *Spectator user* again.

One and only one *Pilot* user per time, during the session, controls the presentation. The *Pilot* is the unique user that can fully interact with the presentation. His or her interactions are captured and sent to all others users' presentations. In fact, the *Pilot user* does not interact only with his or her presentation, but he or she is in the control (or piloting) of all the presentations. The *Pilot user* can also broadcast his or her webcam audio/video to the *Spectator users* and communicate via a text chat.

Besides the *Manager* and the *Pilot user*, all other users engaged in a session are *Spectators*. they have no control of their presentations; they just watch the result of *pilot user*'s interactions. *Spectators* can see the Pilot webcam image/audio (if available) and communicate with other

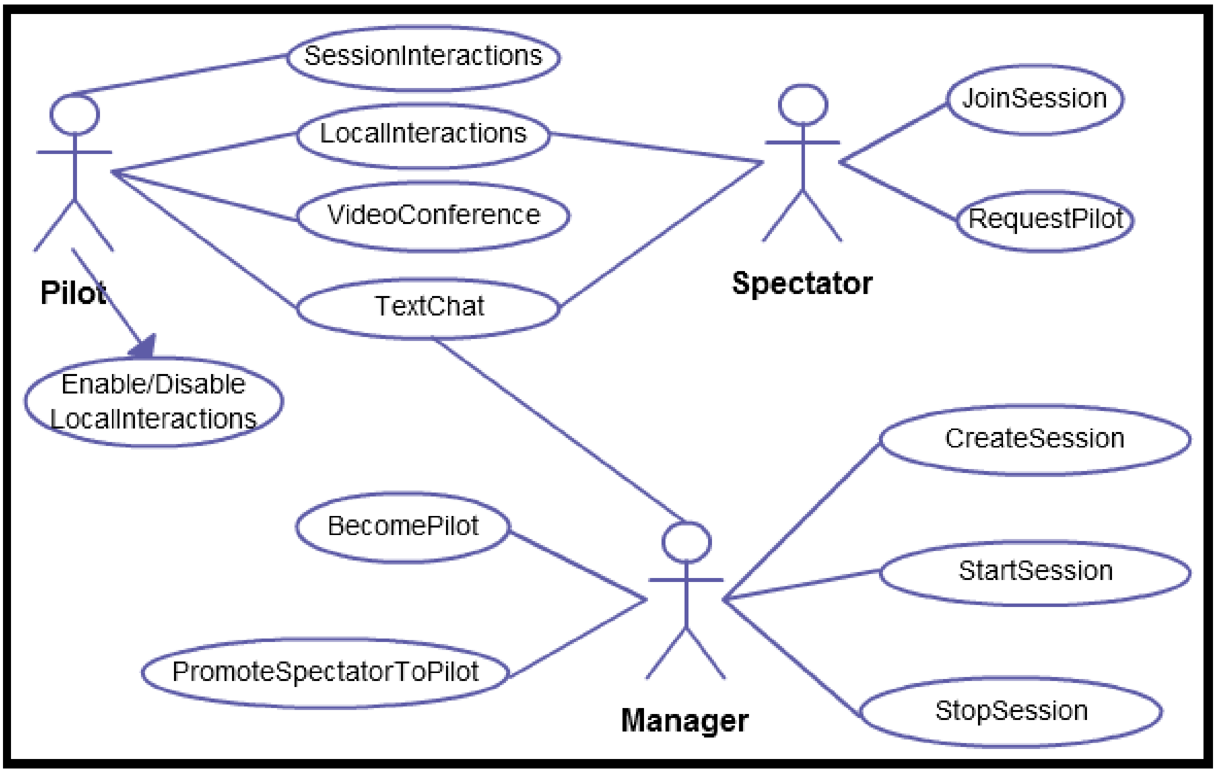
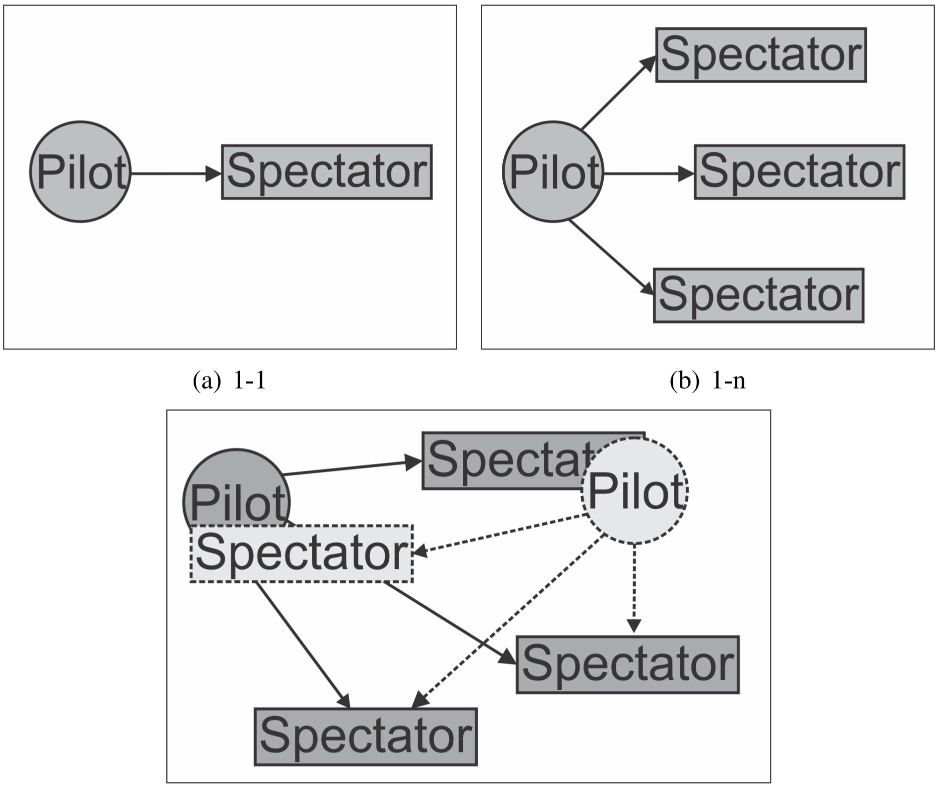


Figura 5.1: User Case Diagram

users via text chat. They can also request to the *Manager user* the permission to become a *Pilot*.

Note that presentations may contains two different interactions categories, named as *Session Interactions* and *Local Interactions*. *Session Interactions* are interactions that affect all the session's presentations (e.g. video seek) and may be performed only by the *Pilot User*. On the other hand, *Local Interactions* only affect the presentation in which the user is locally interacting with (e.g. zooming in an image) and can be performed both by *Pilot* and *Spectator users*. The *pilot* user can enable and disable *local interactions*. The determination of which category an interaction is depends on the application and should be analyzed on a case by case basis.

Figure 5.2 depicts different scenarios in which the Remote Pilotage can be used. A basic scenario is 1-1, in which there are one *Pilot* and one *Spectator user* (Figure 5.2(a)). In the 1-N scenario (Figure 5.2(b)) the *Pilot user* pilots many different *Spectator users*' presentations. In the N-N scenario (Figure 5.2(c)), the *Pilot user* changes over the time, allowing different users control all the presentations, but not at the same time.



(a) 1-1

(b) 1-n

(c) n-n

Figura 5.2: Scenarios

5.4 Challenges

The Remote Pilotage has two main features: (i) the possibility of one user to control the presentation of all other users; and (ii) the users communicate in real-time with each other by a text chat or video conference. The former requires that continuous media, such as video, audio or animation be synchronized; and that media's spatial-temporal relationship, and the result of user's interactions, be consistent among all the presentation instances.

In order to implement (ii), it is necessary a *Low-Latency Communication Channel*, meanwhile (i) requires *Distributed Media Synchronization* and *Inter-media Synchronization*. There is also the necessity of allowing the *Later Join* in a remote Pilotage Session, e.g. allows a user to join in a session that has already started. The remaining of this section describes how these challenges can be overcome with technologies available at the date this paper was written.

5.4.1 Low-Latency Communication Channel

For a low-latency communication channel with video or audio conference, it is necessary the use of low-delay audio and video codecs and decoder. G.279 and Speex are some of the low-delay encoders used in commercial Voice Over Internet Protocol (VOIP) applications. These codecs present low-bitrate and are good for voice encoding, however they present a poor audio quality. High-fidelity low-delay audio codec, such as Mpeg-4 Advanced Audio Coding Enhanced Low-Delay (AAC-ELD) (SCHNELL MARKUS; SCHMIDT, 2008) and CELT[1] (VALIN; TERRIBERRY; MAXWELL, 2009) could also be considered, however these codecs present higher bitrates.

For video, a common approach is to optimize broadly-adopted video codec for low-delay communication, usually with specialized hardware aid. For instance, the Mpeg-4 H.264 Advanced Video Coding (AVC) can be optimized for low-delay communication by reducing its internal buffers or avoid using B frames ((VIEL et al., 2011; KEGEL et al., 2012)). The coded audio and video must also be packaged and transported by some real-time protocol, such as Real-Time Transport Protocol (RTP) or Real-Time Streaming Protocol (RTSP) for IP networks.

Another option is RTMP[2], which was developed by Macromedia and later acquired by Adobe to provide real-time communication between a Flash player and a media server. It supports audio, video and data streaming over IP network such as the Internet. It has also variations that include security, encrypted connections and packets encapsulated within HTTP requests to avoid firewall security.

A video conference during a remote pilotage session with hundreds of users, such as in Massive Open Online Courses (MOOCs), may become unfeasible. A hierarchical structure of tutors may be a solution. Students may be able to see and hear the teachers explanations, but they are allowed to interact only with the tutor designated to help the group in which they are included.

5.4.2 Distributed Media Synchronization

Continuous media, such as video, audio and animation, must be synchronized when multiple users are watching the content together. It means that the videos of each presentation instance of a remote pilotage session should be playing the same frame. Some works report that differences in video's playback up to 1 second do not impact on users' experience, but differences bigger than that should be corrected (GEERTS et al., 2011; VAISHNAVI et al., 2011).

There are three classics approaches to keep continuous medias synchronized: master–slave,

[1] CELT is now part of the IETF Opus Codec http://opus-codec.org/

[2] Real-Time Messaging Protocol

sync-maestro and distributed control. In the master-slave, one presentation instance is elected as master and periodically broadcasts to the other (slaves) instances the current timestamps of its continuous medias. If a slave instance finds out that it is not synchronized with the master, it corrects itself. The sync-maestro approach gathers timestamps from all instances and calculates a guideline timestamp which is then broadcasted back to the instances. In the distributed control, all instances broadcast their timestamps and each instance chooses one instance to be synchronized with.

By using one of the three approaches, the presentation's instances can detect when they are desynchronized and need to correct themself. There are three ways to perform the re-sync: (i) time warp (MAUVE et al., 2004), (ii) waiting sync and (iii) playback speed adjustment. In time warp, the instance instantaneously seeks the continuous medias to the correct time in order to synchronize with other instances. In the waiting sync, the most time-forward instances are paused until the late ones reach them. In playback speed adjustment, the speed playback of the continuous media are slightly accelerated or reduced until the medias become, gradually, synchronized again.

The approach (i) is the easier to implement, but it may impact on users' experience. If the time adjustment is too big, the user may lose some important information in the presentation. The approach (ii) is a global approach and affects all the presentation instances. It may be very annoying to users if one instance lose synchronization and become later often. The approach (iii) may lead impacts on inter-media synchronization and the speed changes may impacts on users' experience.

5.4.3 Inter-media Synchronization

Multimedia presentations usually have spatial-temporal relationship among its medias. For instance, in SMIL or NCL it is possible to specify that when a continuous media reach a certain time, another media must be displayed in the screen. The relationship among medias is also affected by users interactions. For instance, in NCL it is possible to specify that when a button is pressed a media playback must be paused. Although the given examples are quite simple, the spatial-temporal relationship among medias may be very complex, involving many medias and conditions.

One approach that might be used in order to keep inter-media synchronization among different presentations instances is rely on the local presentation machines of each presentation instance. In this approach, hereinafter called **Local Synchronization**, only the unpredictable events (such as user interactions) are sent to spectators' instances.

Other possibility is disabling any formatting and scheduling functionality of the spectators' instances and uses the local presentation machines only as media players. In this approach, hereinafter called as **Centralized Maestro**, not only the unpredictable events are broadcasted from the pilot's instance, but all programmed actions that modify presentation, such as an start or stop action over a media.

The main drawback in **Local Synchronization** is that this approach is more likely to result in inconsistencies among the presentations, especially due to corrections of distributed media synchronization. Suppose, for example, a program that imposes to a video a resize when it reaches 20s. In a 5s time-forward spectator instance the video would reach 20s and would be resized, but the the correction to synchronize it with the other session's presentations, using the time wrap approach, would move the video to 15s but would keep its new dimensions.

In the **Centralized Maestro**, all media behavior and playback are controlled by a single presentation machine, so it is possible to avoid some inconsistencies caused by the local presentation machines. In addition, since the spectators' instances do not need to handle the inter-media synchronization by themselves, they can be simpler than the pilot's presentation machine. Note that this may prevent some users to become pilots.

One drawback of **Centralized Maestro** is that it may flood the network with lots of control messages. It also may be ineffective in scenarios in which the network delay is high. For instance, if the network delay between the pilot's and a spectator's machine is 1s, the control message that would make the video to be resized at the second 20, from the previous example, would only reach its destination in second 21.

A solution for this delay problem is the employment of the **local lag** technique (MAUVE et al., 2004). In local lag, user's interactions performed locally are only resolved after a certain time; usually the estimated network latency. However, for time-based events, such as the video resize at second 20, it is not enough. It would only make the video be resized in the second 21 in both pilot and spectator's presentation. In order to perform the resize in second 20, it requires a prediction engine in the pilot's instance that send the control messages before the event occurs.

5.4.4 Later Join

For a user to join a remote pilotage session after it has already started, it is necessary to know the current state of the other presentations instances. The state of an interactive multimedia presentation can be associate to a node in a temporal graph, such as in SMIL State (JANSEN; BULTERMAN, 2009) or NCL Eventline (NETO et al., 2012).

A possible solution to allow later join is to keep the current state of the presentation of the temporal graph in the pilot's instance. When a new user joins the presentation, the current state is informed to his or her presentation machine and, by the temporal graph, it restores the presentation state. The problem of this solution is that temporal graph tends to become very complex, especially in highly interactive presentations, and it is not a trivial task restore the presentation state from the graph's state in another presentation machine.

Instead of keep the current state, it is possible to store the events that lead to transitions in the temporal graph, as an event stream. When a new user joins the presentation, the event stream is sent to its presentation instance and it reproduces the events in order to achieve the current presentation state. Note that some optimizations in the event stream can be performed. For instance, if there are two seeks in a video, only the second one needs to be carried out.

Rather than consider the presentation state in whole, we can consider each media's state individually. Each media has a number of properties, such as spatial position, playback state (playing, paused, stopped, etc.) and, for continuous media, current playback time. It is possible, when a user joins the session, dump all the media's properties from the pilot's presentation and restore them into the new spectator instance.

Another solution is splitting the presentation into independent sub-presentations. A sub-presentation is not affect by any interaction performed by users in the other sub-presentations. The state in the temporal graph in which a sub-presentation starts is predictable and can be reproduced easily, no matter if it is a presentation that is already engaged in a remote pilotage session or a new presentation instance that just joined. These states are like milestones in the temporal graph. In this approach, when a new user wishes to join an already started session, it should wait until the beginning of next sub-presentation to engage the activity.

All the aforementioned solutions for later join can be implemented by the presentation machine itself and they work regardless of the multimedia presentation (although the sub-presentation approach expects a certain presentation organization). It is also possible to conceive application-bounded solution for later join. The presentation's state would be defined by a couple of well-known variables. When a new user joins the session, his or her presentation machine would receive the value of these variables and the associate application would restore the state. The application-bounded solution may be easy to be implemented for most multimedia presentation, however it requires imperative code. It may be tricky to implement in declarative synchronization languages, such as SMIL and NCL.

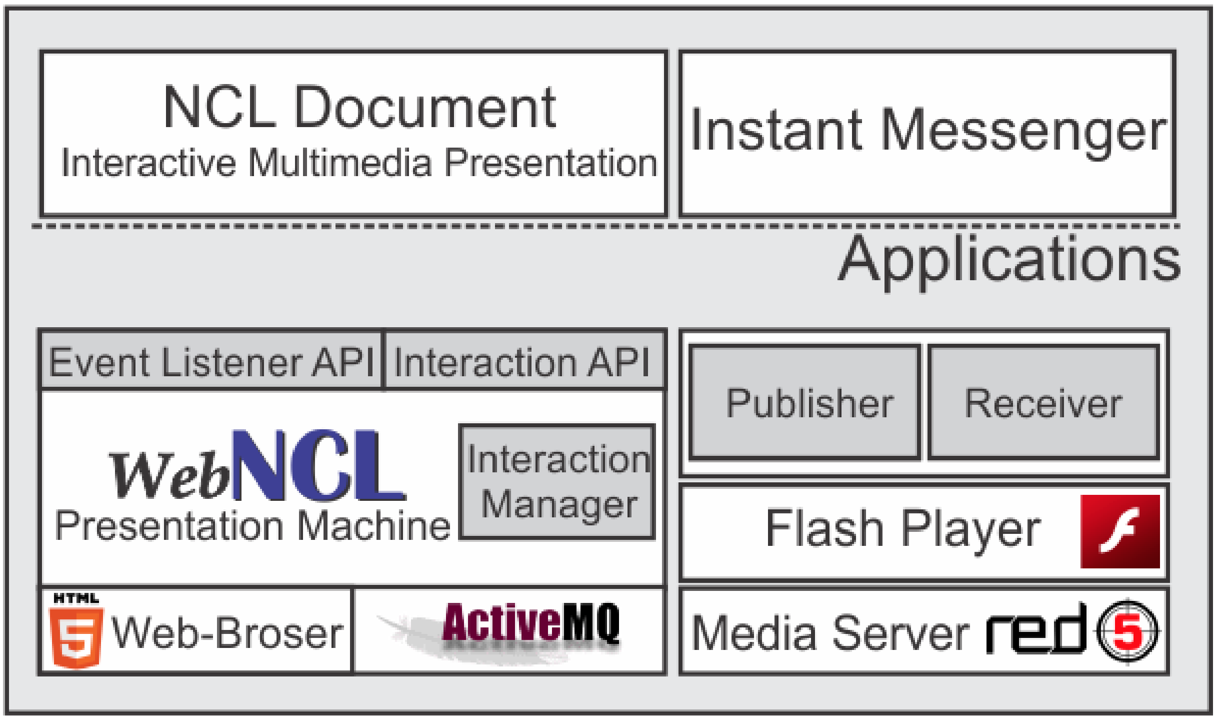


Figura 5.3: Proof-of-Concept Architecture

5.5 Proof-of-Concept

We have instantiated the Remote Pilotage approach for NCL documents. Our focuses in the proof-of-concept was the exploration of inter-media synchronization and later join. We have also implemented a low-latency instant messenger with support for text chat and video-conferencing based on Adobe Flash platform. Figure 5.3 depicts an overview of the proof-of-concept.

We have extended the WebNCL (MELO et al., 2012), a NCL presentation machine based on the Web-stack (HTML5/Javascript/CSS3). The extension consists of a new API, the Event Listener API, in which a function can be registered for listening to specific events that may occur in a NCL presentation, such as input events or media events (starting, pausing, etc.). We also extended the Interaction API (the API from which WebNCL receives input events, such as from virtual remote control) to add the passive mode. In this mode the user cannot interact with the presentation, e.g. events detected by the WebNCL's Interaction Manager are ignored.

Listing 5.1 illustrates the use of the extended WebNCL's API. In Lines 10-12 the WebNCL is instantiated; in Lines 13-17 a function for listening to *input* (such as focus changes and key pressed) and *presentation* (high level-events, such as the beginning and the end of the presentation) events is registered. In line 18, the passive mode is enabled in the WebNCL instance, which means the player will ignore any local user's interactions.

Listing 5.1: WebNCL’s API using

```
<html>
<head>
    <script type="text/javascript"
            src="webncl.deb.js"></script>
    ...
</head>
<body>
    <div id="playerDiv"></div>
    <script>
        var player = new WebNclPlayer(
                "main.ncl",
                "playerDiv");
        player.addListener(
            function(evt) {
                console.log(evt);
            }, [’input’,’presentation’]);
        };
        player.setPassiveMode(true);
    </script>
</body>
</html>
```

By using these APIs we implemented the Inter-Media synchronization, based on the Local Synchronization approach. Registers for input and presentation events are done in the pilot’s instance. These events are encoded in JSON and sent to the spectators’ presentations by means of an ActiveMQ [3] message broker via the stomp protocol over a websocket. The spectator’s presentations are all in the passive mode, but when they receive the input events from the pilot’s presentation via broker, they post events using WebNCL’s input API.

The control messages are also encoded in JSON and sent via broker to all presentation machines. For example, when a spectator instance is promoted to pilot, we register a function for listening events in the new pilot, setting the passive mode to false. The previous pilot’s presentation passive mode is set to true and it becomes a spectator instance.

We have implemented the later join using the sub-presentation approach. The NCL docu-

[3] Apache ActiveMQ - http://activemq.apache.org/

ment must implicitly declare which are its sub-presentations. A sub-presentation must be an NCL <context> children from the <body> node with the anchor property *isMilestone* set as *true*, as illustrated in Listing 5.2. When a <context> node, which is a sub-presentation, starts, a presentation event is sent by the pilot's presentation. All the presentation instances, that are waiting, start when a sub-presentation begins. But instead of starting from the <body> context node, they start from the sub-presentation <context> node.

Listing 5.2: Sub-Presentation as <context> node

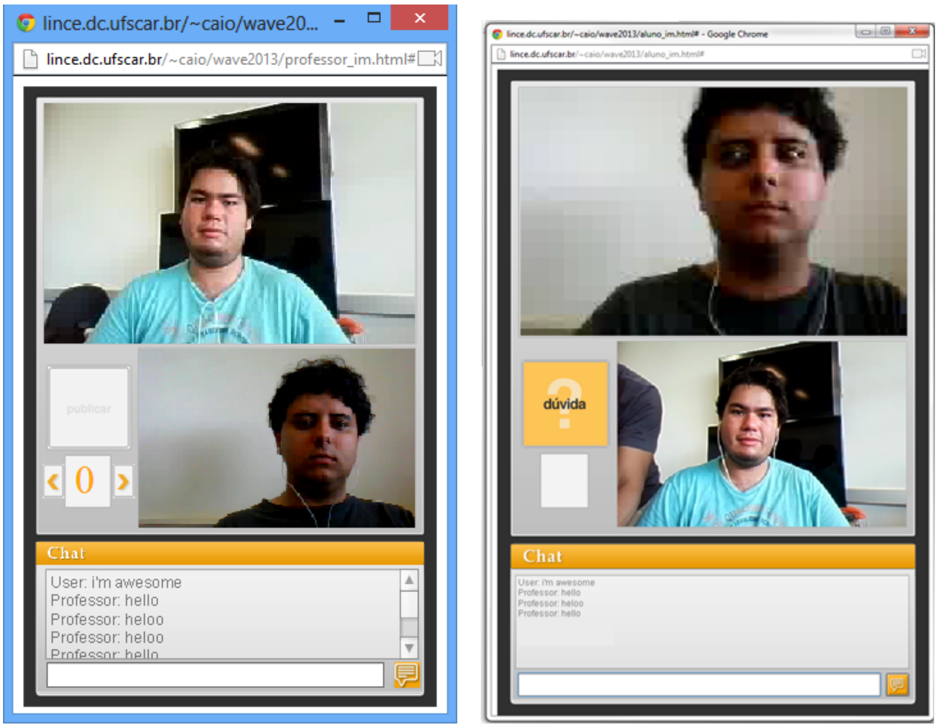
```
<ncl>
<body>
    <context id="subpresentation1">
        <property name="isMilestone"
            value="true" />
        ...
    </context>

    <context id="subpresentation2">
        <property name="isMilestone"
            value="true" />
        ...
    </context>
</ncl>
```

The instant communication is developed using flash components developed in the scope of Tidia-Ae project (GASPAR; PRADO; TEIXEIRA, 2009). The media server used in this proof-of-concept is the open source Red5 project [4].

The audio and video real-time communication of the instant messenger needs a publisher and one or more receivers. The connection between the publisher and a receiver is accomplished by the *streamID* created by the publisher when publishing a streaming. The publisher sends audio and video data encapsulated in RTMP protocol to the media server. The server is responsible to distribute data streams to each subscribed receiver. The developed proof-of-concept creates a combination of publisher and receiver, so both the pilot user and the spectator users can send and receive audio and video.

In our proof-of-concept, only the pilot user may publish his video at the beginning of the

[4] Red5 Media Server - http://www.red5.org/


lince.dc.ufscar.br/~caio/wave20...
lince.dc.ufscar.br/~caio/wave2013/professor_im.html#
publicar
0
Chat

User: i'm awesome
Professor: hello
Professor: heloo
Professor: heloo
Professor: hello

lince.dc.ufscar.br/~caio/wave2013/aluno_im.html# - Google Chrome
lince.dc.ufscar.br/~caio/wave2013/aluno_im.html#
dúvida
Chat

User: i'm awesome
Professor: hello
Professor: heloo
Professor: hello

(a) Pilot User

(b) Spectator User

Figura 5.4: Instant Messenger Interface

session. A spectator user must require the permission to publish his or her video. If a pilot or manager grants to the spectator user the permission for publish his or her video, then the entire session will receive the spectator user's stream. Text chat, however, is always available for all users.

Figure 5.4 depicts two instances of the instant messenger interface when a spectator user is publishing his or her video. Figure 5.4(a) is the pilot view and Figure 5.4(b) is the spectator view.

5.5.1 Using the Remote Pilotage

We have used the Remote Pilotage in order to control complex multi-video and highly-interactive NCL document automatically generated from the capture of a lecture-style live-presentation. The capture and generation process of these presentation are described elsewhere (VIEL et al., 2013c).

These multimedia presentations are compounded by four synchronized video streams. The user may select which video he or she wishes to see in detail in a bigger window. He may also navigate in the presentation using the timeline or by points of interest, such as slide transitions.

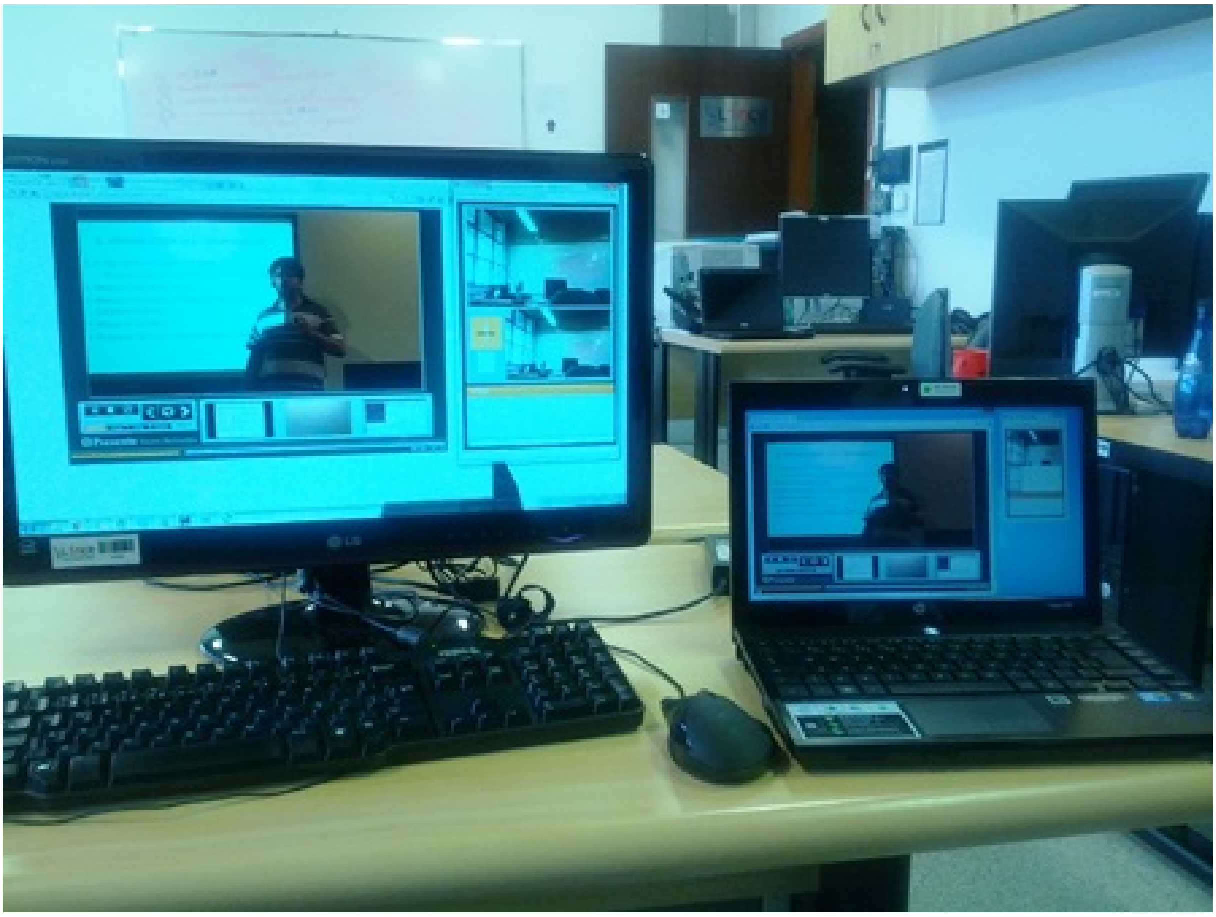


Figura 5.5: A Remote Pilotage Session

Figure 5.5 shows a picture with two presentation instances. The presentation at the right (notebook) is the pilot instance and it is controlling the left one (desktop), which is a spectator instance. All the events performed in the pilot instance, such as choosing the main video or navigating in the presentation, were reproduced with fidelity in the spectator instance; keeping the presentations synchronized.

In order to validate the proof-of-concept prototype, we have evaluated: (i) the coherence among the multimedia presentation instances; (ii) how long an interaction performed in the pilot's instance takes to affect the spectators' instance; (iii) audio and video quality of the video conference; and (iv) the communication latency.

We tested the remote pilotage proof-of-concept in four different scenarios: (i) LAN with a local broker and media server; (ii) LAN with a remote broker and media server; (iii) users geographically distant with the broker and media server hosted in the pilot super node; and (iv) users geographically distant with the broker and media server hosted in a cloud.

In our tests, we have used computers with enough processing power to present the multi-video multimedia presentation smoothly (such as Core 2 Duo 2.6 Ghz and 4 GB of memory or

Scenario	Coherence	Interaction Delay	A/V Quality	Comm Latency
LAN; servers local	Yes	Unnoticeable	High	Unnoticeable
LAN; servers remote	Yes	Yes, < 500s	Low	Yes, < 500ms
Distant nodes; servers in pilot	Yes	Yes, < 300s	Low	Yes, < 300ms
Distant nodes; severs in a cloud	Yes	Yes, < 300s	Low	Yes, < 300ms

Tabela 5.1: Tests in Different Scenarios

superior). They were also connected through a broadband network.

Table 5.1 summarizes the results. In scenarios (i) and (ii) we have used up to 5 computers running an instance each. In scenario (i) we have not seen any noticeable incoherence or interaction delay in the instances. We could set audio and video quality to high and we did not notice latency in communication. In scenario (ii), the instances were located in São Carlos - SP, Brazil and the broker and media server were located in an Amazon Server in Ireland. Although no coherence losses were noticed, a delay up to 500ms between the pilot and the spectators was observed. The quality of the audio and video was reduced to achieve a low latency in the video conference.

Scenarios (iii) and (iv) were tested with a node located in São Carlos - SP, Brazil, a node located in San Diego - CA, USA, and a node located in Corfu, Greece. In scenario (iii) the broker and the media server were located in Brazil, and the node in Brazil was the pilot. In Scenario (iv) the broker and media server were located in a Amazon Server in Ireland. We did not notice any coherence loss in both scenarios, but there was a delay up to 300 between the pilot and the spectators. The quality of the audio and video was reduced to achieve a low latency in the video conference.

5.6 Final Remarks

Synchronized distributed multimedia consumption is a facility which may enable the conception of interesting applications. When the interaction can affect the content's playback, only the synchronization of continuous media is not enough. The Remote Pilotage approach, presented and evaluated in this paper, is an effective contribution to reach a solution for such facility.

We presented a proof-of-concept that implements the Remote Pilotage model for NCL documents. The instant messenger is built using flash components. The inter-media synchronization is based on Local Synchronization approach and Later Join are enabled by Sub-Presentation strategy.

In our tests the video conference presented a low latency, but at the cost of audio and video quality. Other technologies for real-time communication, in special the WebRTC API, a HTML5 emerging feature that enables video-conferencing in the Web without the aid of plugins, are going to be investigated in the continuity of this work.

In our study cases, we have noticed that users' communication, both by text chat and audio/video, really improves the experience when interacting with multimedia presentations. As a future work, we intend to investigate means for enriching the multimedia presentation with the interactions performed by the users during a remote pilotage sessions, as suggest by the works of Cattelan et. al. (CATTELAN et al., 2008), Pimentel et. al. (PIMENTEL; JR.; CATTELAN, 2007) and Mullher and Ottman (MüLLER; OTTMANN, 2000).

Capítulo 6

How are they watching me: learning from student interactions with multimedia objects captured from classroom presentations

The performance of a teacher in the exposition of a subject is a rich experience that can be captured and transformed into a corresponding multimedia learning object, given the multimodal and multi-device nature of the presentation. Using as a starting point an interactive multimedia object which is an electronic version of a problem solving lecture recorded by the teacher, in this paper we report how a group of students interacts with one multimedia learning object composed of synchronized videos, audio, images and context information. The qualitative analysis of the data allows the teacher to infer useful information not only for refining the lecture content but also for improving its presentation. The case study presented illustrates how a similar analysis can be performed by other instructors with respect to their own lectures, and demonstrates both the power of capturing the multimodal and multi-device nature of the original presentation, and the utility of logging the student-multimedia learning object interaction.

6.1 Introduction

When lecturing to her students, the performance of an instructor in the classroom can be considered a multimodal and multi-device live presentation that can be captured and transformed into a corresponding multimedia learning object.

The classroom activity is the primary learning context in many courses (ABOWD et al., 1999), so capturing such activities, lectures in special, may be interesting for several reasons. From the attendee's perspective, a student may use the recordings when solving assignments or to study for an exam, or a student who misses a class may still have access to what was presented by watching the recordings. From the instructor's perspective, a professor who will be absent from the campus may prepare a recorded lecture to deliver to the students. Moreover, a previously captured lecture may be improved and reused, or a portion of captured lecture may be used as a complementary learning object in different educational approaches. Last but not least, captured lectures can be a valuable resource for e-learning and distance education courses (LIU; KENDER, 2004).

We are aware that there are strong divergences among educators as to the efficiency of the lecture format as a method of instruction in middle school and higher education. Ross, for example, states that "when I was younger, I used to say that it took 40 years for any change in significant higher education to take effect, because that was the time by when all the existing teachers would have retired. I now realize that I was not a cynic, but an optimist, since lectures are just the prevalent as they ever were" (ROSS, 2011). However, as Ross himself acknowledges, lectures are still widely used in all levels of education. Moreover, Schwerdt and Wuppermann observe that "contrary to contemporary pedagogical thinking, we find students score higher on standardized tests in the subject in which their teachers spent more time on lecture-style presentations than in the subject in which the teacher devoted more time to problem-solving activities" (SCHWERDT; WUPPERMANN, 2011).

Although recording lectures is common practice in several universities, producing quality video lectures demands a high operational cost. To reduce such costs, many tools for the (semi) automatic capture of lectures were developed in the past (BROTHERTON; ABOWD, 2004), (CHOU et al., 2010), (DICKSON et al., 2010), (HALAWA et al., 2011), (NAGAI, 2009). However, such tools usually record only video streams and generate, as a result, a single video stream (e.g. a podcast). In several scenarios, this may not be always enough to reproduce the classroom experience.

The classroom itself can be viewed as a rich multimedia environment where audiovisual information is combined with annotating activities (ABOWD et al., 1999). Furthermore, the context of the class (e.g. the slide being presented, what the lecturer says and her body language) and how the different audiovisual contents relate to each other are also important. For instance, sometimes it is necessary to relate the slide presentation with the whiteboard for the comprehension of an exercise or lesson (DICKSON et al., 2012). In addition, the interaction between the lecturer and the students is also a valuable part of the learning process.

In this work, capturing a presentation means recording the audio and one or more video streams of the speaker, the images presented on the screen or projector, the writings and drawings made on whiteboards, and capturing relevant contextual information – the aim is to use the captured information to automatically generate an interactive multimedia object, as proposed by the Linking by Interacting paradigm (PIMENTEL; ABOWD; ISHIGURO, 2000). We refer to as an "interactive multi-video object" the composition of several videos, audio and some static media, properly synchronized and with facilities for flexible interaction and browsing.

From the multi-video object, the lecture may be reconstituted and explored in dimensions not achievable in the classroom. The student may be able, for example, to obtain multiple synchronized audiovisual content that includes the slide presentation, the whiteboard content, video streams with focus on the lecturer's face or the lecturer's full body, or the lecturer's web browsing, among others. The student may choose at any time what content is more appropriated to be exhibited in full screen. The student may also be able to perform semantic browsing using points of interest like slides transitions and the position of lecturer in the classroom. Moreover, facilities can be provided for users to annotate the captured lecture while watching it, as suggested by the Watch-and-Comment paradigm (CATTELAN et al., 2008).

In this paper we report how a group of students interacts with a multimedia learning object composed of synchronized videos, audio, images and context information, and discuss how the analysis of the interaction data allows the instructor to infer useful information for improving the lecture. The case study illustrates how a similar analysis can be performed by other instructors with respect to their own presentations, and demonstrates both the power of capturing the multimodal and multi-device nature of the original presentations, and the utility of logging the student-multimedia learning object interaction.

This paper is organized as follows: in Section 6.2 we discuss related works; in Section 6.3 we describe our proposed model to capture live lectures; in Section 6.4 we present our current prototype implementation; in Section 6.5 we present one case study in which one instructor used the prototype to capture one problem solving session and generate an associated multimedia learning object; in Section 6.6 we detail lessons learned from the instructor after a qualitative analysis of the interaction a group of students had with the learning object; and in Section 6.7 we present our final remarks.

6.2 Related Work

Several authors report results from building systems designed to capture lectures. The AutoAuditorium records classroom activities using a spotting and a tracking camera controlled by computers. The camera orchestration is carried out in real-time using some heuristics based on audiovisual production. The main idea is to create a “TV-like” production without the usual cameraman, video director, audio engineer and other professionals (BIANCHI, 2004).

Lampi et al. consider the use of multiple cameras to record lectures. The authors use sensors and computational vision techniques to do the cameraman’s job. They also use a finite state machine to define, at each moment, which camera stream should be included in the final stream (LAMPI; KOPF; EFFELSBERG, 2008).

Nagai uses an environment with a high definition camera (*Advanced Video Coding High Definition* - AVCHD) placed at the back of the classroom. The camera can record the whole lecture scene (lecturer, whiteboard, slide presentation, students, etc.). By using tracking techniques, the camera performs digital zoom to what is considered the focus of attention at different moments (NAGAI, 2009).

Chou et al. use tracking techniques to detect the lecturer’s movements and screens (whiteboard, slide presentation) changes. A camera action table is then queried to get what must be done (zoom in, zoom out, pane, etc.) in order to highlight the image that must be the focus of attention (CHOU et al., 2010).

All the aforementioned works differ from the work reported in this paper in that the resulting product of the lecture capturing process is a single video stream instead of a multi-video object.

In the work of Liu et al., lectures are captured in a similar process to the ones mentioned before, resulting a single video stream. The difference is that the set of slides used in the presentation is added to the video stream. However, the slides are not synchronized with the video (LIU; KENDER, 2004). Given that the result is single-video-stream, students do not have autonomy to choose the camera that gives them the best view of the lecture for each situation, or to focus their point of interest, as allowed in our multi-video object.

ClassX is a tool designed for online lecture delivery (HALAWA et al., 2011) (PANG et al., 2011). A live lecture is captured by means of an AVCHD stream split in several virtual standard resolution cameras. By using tracking techniques, the most appropriated virtual camera for a given moment is chosen and streamed to the remote students. The students have the opportunity to choose a different stream from another virtual camera or even watch the original AVCHD stream, and a synchronized slide presentation is offered — but no other navigation facilities are available the students.

REPLAY is a system for producing, manipulating and sharing lecture videos (SCHULTE; WUNDEN; BRUNNER, 2008). Besides offering similar features to the aforementioned systems, REPLAY uses computer vision to recognize written words, and deploys MPEG-7 to index the videos. Although REPLAY allows more navigation alternatives than the previous systems, it does not produce an independent multi-video object.

Other authors report the use of other features such as image processing and audio transcription (DICKSON et al., 2012), (DICKSON et al., 2010)), (BROTHERTON; ABOWD, 2004), (CATTELAN; BALDOCHI; PIMENTEL, 2003), the result being hypermedia documents that offer interfaces providing different ways of indexing the recorded information. The model for capturing and recovering lectures presented in this paper allows more flexibility. This flexibility results from the ability to specify the context information that must be captured, and to specify how this context information should be combined to generate a multi-video object, or to promote live interventions in the classroom during the capture process — for example in the case that there is a change in the illumination of the room because the light was off.

6.3 Ubiquitous Capture and Authoring

In order to produce quality lecture videos, the conventional lecture recording process usually requires the presence of audiovisual professionals. Our infrastructure offers a self-service approach, allowing the instructor to record a lecture herself. Some solutions usually rely on computational vision, tracking techniques and sensors to perform camera orchestrations in a attempt to produce a single video or audio stream output.

As detailed elsewhere, the model we have proposed goes a step further (VIEL et al., 2013c). As depicted in Figure 6.1, the model aims at capturing all the content presented in the classroom. The capture process is pervasive, does not rely on human mediation and generates automatically an interactive multi-video object which preserves as much as possible of the lecture content and context.

An environment, usually a classroom, is instrumented with physical devices (Figure 6.1(1)), such as video cameras, microphones, whiteboards, interactive whiteboards and slide projectors. The instrumented classroom may also contain sensors, such as temperature sensors and luminosity sensors, and secondary screens, such as notebooks, TVs, tablets, etc. The video cameras should be placed in points where they can frame important classroom's points (instructors, stu-

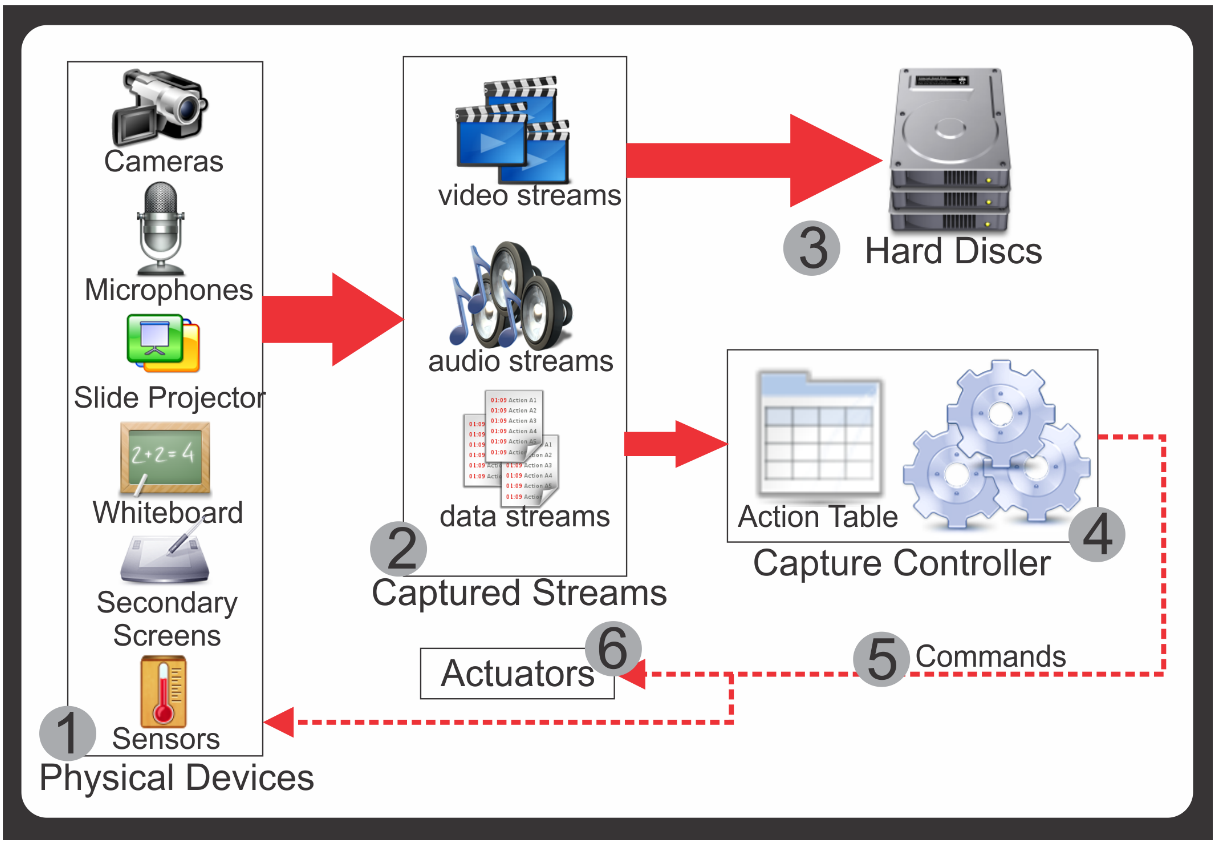


Figura 6.1: Capture Workflow

dents, whiteboard, slide presentation, etc.).

Computer devices capture all the content produced by the physical devices used in the classroom (e.g. whiteboards and slides) and represent them as video, audio and data streams (Figure 6.1(2)). Cameras produce video and audio streams, microphones produce audio streams and sensors produce data streams. By capturing the screen output from the secondary screens or by intercepting the signal sent to the slide projector, we can also produce video streams. The electronic whiteboard can produce both data and video streams. By capturing its strokes we can generate a data stream; intercepting the signal sent to its projector, we can generate a video stream.

All such streams are stored (Figure 6.1(3)) for further use in the multi-video object generation. The streams are also sent to the *capture controller* (Figure 6.1(4)), a component responsible for managing the capture process. The *capture controller* uses signal analysis to analyse the captured streams and to send commands (Figure 6.1(5)) back to the physical devices and actuators (Figure 6.1(6)) present in the classroom.

The instructions in the *capture controller* are defined in a customizable action table. The action table can be used to define actions for certain events which may occur during the capture process. For instance, zooming into the image of a specific camera when the lecturer starts talking, or activating an actuator in order to reduce the light intensity when the lecturer starts a slide presentation.

Our model allows the instructor to split her presentation in different modules, an approach usually adopted in e-learning platforms.[1] A multi-video presentation can be composed of one or more modules. This is useful to better organize the content of a lecture. The lecturer may, for instance prepare a problem solving presentation with one exercise per module. And, it also allows the lecturer to take breaks during the recording process and the students to navigate in the modules of the multi-video presentation.

Splitting the presentation into modules can also minimize the time need for repeating the recording in case of errors. For instance, if in one module the lecturer starts stuttering or becoming confused and wishes to make a retake, she only needs to record that module again. Reusing the modules to compose a new presentation is another advantage of splitting the recording process into modules — reuse is in fact one of the main ideas underlying learning objects.

Given that the processes of analysing and converting the captured streams can demand much computational power and time, once the capture process is finished the data is transferred to a server for further processing.

Considering points of Interest as moments in the lecture which may have particular importance for students, we designed recognizer components that use one or more captured streams to automatically detect potential points of interest. The points of interest can be used to provide a more semantic navigation over the multi-video object, allowing the students to seek for the next slide transition, for instance.

Some points of interest have been suggested in the literature ((DICKSON et al., 2012), (CATTELAN; BALDOCHI; PIMENTEL, 2003) and (BROTHERTON; ABOWD, 2004)), while others were inspired on our own observation of real lectures. Examples of Points of interest are slide transition, whiteboard interaction and change the eye-gaze of the instructor.

The resulting multi-video learning object is composed of videos and other captured media. Although the multi-video object cannot reproduce several aspects of the live lecture experience (live interactions, odors, temperature, etc.), it offers other facilities to the students when they are interacting with the object.

[1] Examples include http://www.coursera.org and http://www.edx.org

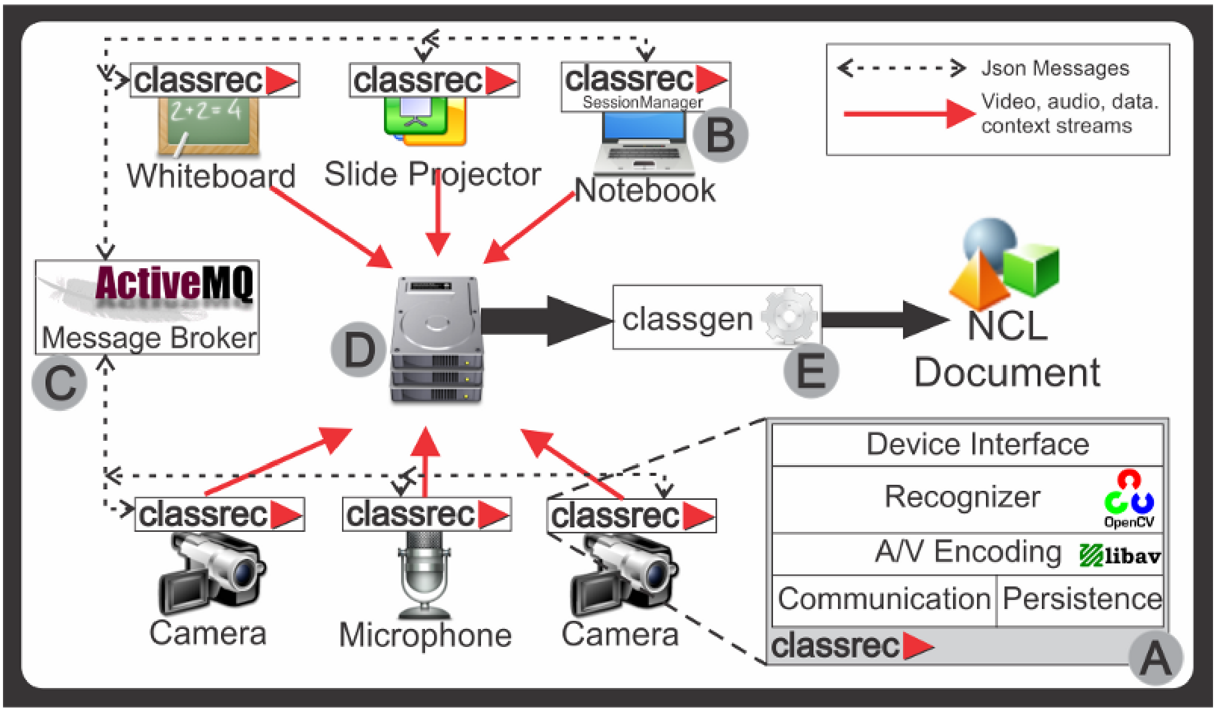


Figura 6.2: Prototype Overview

6.4 Prototype

As a proof-of-concept of the model, we developed a prototype tool for capturing lectures and generating multi-video objects. This prototype was mainly developed in Python. Figure 6.2 depicts an overview of the prototype.

The prototype is composed of three main parts: the *Capturing tool* used to capture streams; the *Processing tool* in charge of stream analysis and the generation of the multi-video object; and the *Presentation tool*, which allows the user to playback the multi-video object.

6.4.1 Capturing Tool

The *Capturing tool*, named *Classrec*, (Figure 6.2(A)) performs the lecture capturing process. Each computer used in the capturing process runs an instance of *Classrec*, and one of these instances is selected to be the session manager (Figure 6.2(B)). It corresponds to the *Capture Controller* of the workflow (Figure 6.1). The session manager is responsible for handling the lecturer's stimulus and for controlling the other *Classrec* instances, keeping them synchronized.

The capturing process is based on video streams. *Classrec* captures content (video and audio streams) produced by AVCHD and outputs produced by computers (such as computer screens, slide presentations, etc.). It also records metadata about the lecture, such as module structure, available streams and authoring information into an XML file.

We opted to capture the electronic whiteboard output as a video stream instead of its strokes. This was done because a video stream is more portable than strokes and, given the modern video encoding as h.264 Advanced Video Codec and the static nature of whiteboard outputs, the bit rate of the video stream is low. We could record a stroke stream, but it would require a specialized media player to play it back (as it is the case with other systems, e.g. (MüLLER; OTTMANN, 2000)).

Some streams, such as slides, whiteboards and computer screens may contain segments with a lot of static content, but they are still captured as video streams. A possible improvement would be to replace the video for a combination of non-static content videos and a single image to represent a static segment (video with no changes during a period of time).

The communication among the different applications is carried out using the Apache ActiveMQ message broker (Figure 6.2(C)).

6.4.2 Processing Tool

The *Processing tool*, named *Classgen* (Figure 6.2(E)), performs the multi-video generation process. This tool uses as input the video streams and metadata recorded by *Capturing tool*. It also supports an XML configuration description language, which allows the specification of which recognizers (and its inputs) should be used, and the codecs that should be used to encode audio and video.

We have implemented recognizers capable of detecting (i) the presence of a lecturer in a video stream; (ii) if the lecturer is facing a camera; (iii) slides transitions; (iv) interactions with whiteboard or PC; and (v) a list of spoken keywords.

It is also possible to specify an orchestration of video streams in order to produce a new video stream. This is useful in environments with multiple cameras recording different angles of the lecturer. Through the XML configuration description language, it is possible to select which stream will be used in the orchestration and how to orchestrate then. For instance, it is possible to specify that when a recognizer detects the lecturer's face in video segments, the camera orchestration stream should include that segment.

Classgen uses the OpenCV library (BRADSKI, 2000) to perform pattern recognitions in order to identify points of interest for composing the context stream. The media manipulation during the orchestration process and the audio/video conversion is handled by the *libav* library.

Once the several processes associated with recognition of points of interest, orchestration and video conversion are concluded, the information they generate (the specification of the points of interest, the orchestration stream, and the converted streams) are stored in the *XML lecture*. The XML is then passed to a component of the *Processing tool* responsible for generating the final multi-video object (Figure 6.2(5)). Our prototype generates NCL[2] (ABNT, 2007) documents, but the *Classgen* can be extended to generate other types of multi-video objects, such as HTML5 pages or stand-alone desktop, tablet or smartphone applications.

The XML configuration description language can also describe the video streams (including the orchestration, if any) and points of interest will be used in the final multi-video object. It is also possible to generate different multi-video objects using the same recorded lecture (for instance, by using the orchestration stream or not).

6.4.3 Presenting Tool

It is desirable to offer students a platform-independent way to access the captured lectures. We would like to avoid students having to install specific software to playback the lecturers. To fulfill this requirement we choose a web-based implementation.

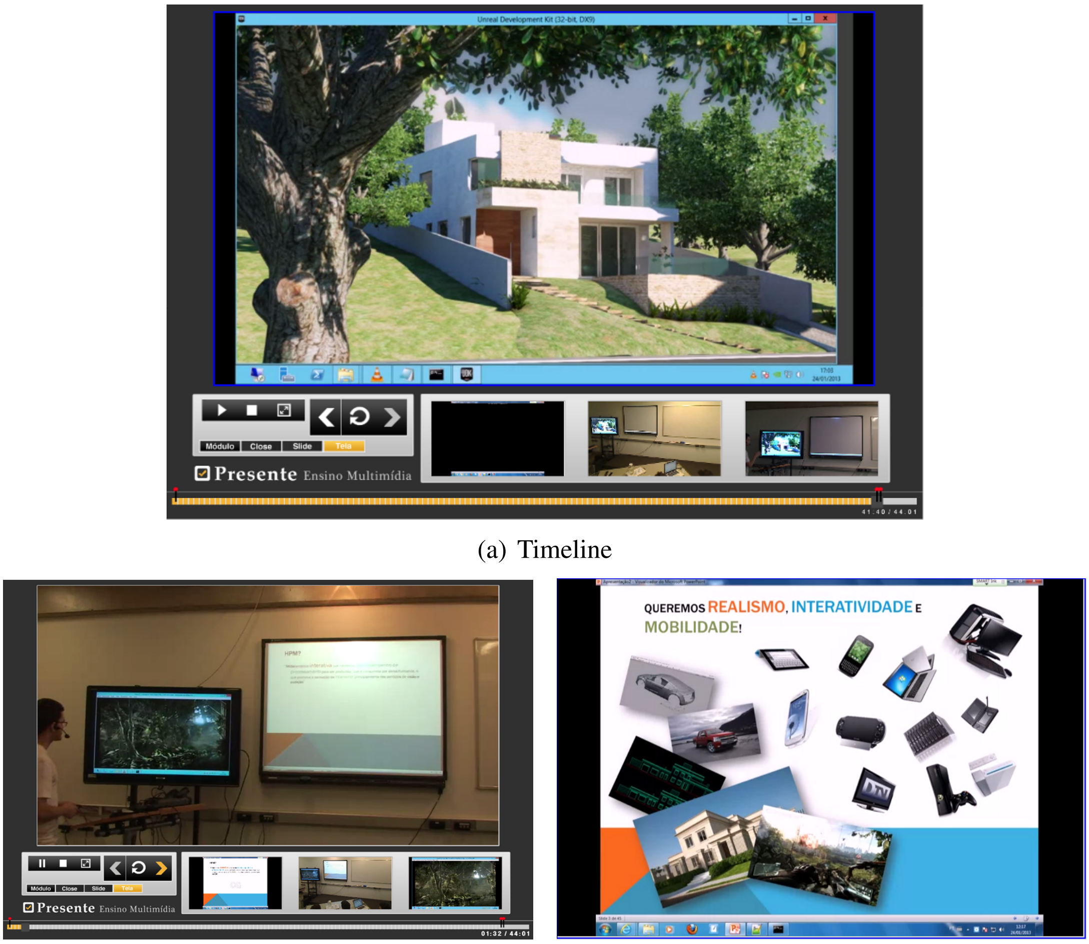
The multi-video object generated from the capture imposed some challenges. In the scenario where we considered the generation of the object directly in HTML5 + JavaScript, a large development effort to implement the synchronization capabilities was estimated. We also noticed that most obstacles identified in the HTML5-based implementation would be easily overcome with the use of a declarative language specialized in media synchronization. However, there were no solutions to support it that did not demand external plug-ins.

As a result of these needs, we were motivated to propose and develop a multimedia presentation engine based on standard Web technologies. We conducted an implementation based on HTML5 + JavaScript that enables the presentation of multi-video NCL documents, named WebNCL[3] (MELO et al., 2012). Thanks to WebNCL, any device which has an HTML5-compatible browser (PC, Smart TV, Tablet, Smart Phone, etc.) can present NCL documents natively.

The choice for implementing support to the NCL language was taken because it is a powerful language for media synchronization, under active development and adopted as iDTV (ABNT, 2007) and IPTV standards (H.761, 2009). A good side effect of this choice was the possibility

[2] Nested Context Language - http://ncl.org.br/en

[3] WebNCL is an open-source software, available at http://webncl.org


Unreal Development Kit (32-bit, DX9)
Módulo
Close
Slide
Tela
Presente Ensino Multimídia
41:40 / 44:01
01:32 / 44:01
QUEREMOS REALISMO, INTERATIVIDADE E MOBILIDADE!

(a) Timeline

(b) Multiple Videos

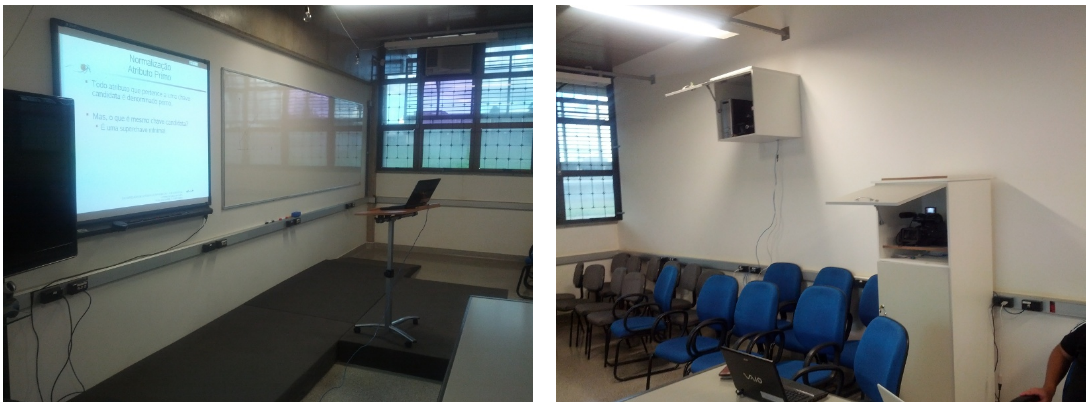
(c) Full-screen

Figura 6.3: Multi-video learning objects

to reuse the content generated in different platforms.

Figure 6.3 shows running NCL learning objects generated by the prototype. The NCL document offers some facilities for students. One of these facilities is the synchronization of the captured audio/video. The multi-video object synchronizes the multiple audio/video streams, so students can see what was written in the whiteboard when the lecturer points to the slide presentation. This synchronization is essential to recover the whole audiovisual context of the captured lecture at a given moment. It is also possible to insert non-synchronized complementary media to the multi-video object like, for instance, an image from a textbook.

The multi-video object offers a more semantic and easy way to navigate in the captured lecture than timeline navigation, common in video (however, timeline navigation is still present). For instance, the student can move forward to the next slide transition or backwards to the previous one. When the lecturer begins to write something in the whiteboard, the student can skip all the writing process and see the final result. In a future implementation, students



(a) Front Side

(b) Back Side

Figura 6.4: Instrumented Classroom

will also search for a keyword and move forward in the multi-video object to the point where the lecturer said "for instance".

Similar to in-classroom lecture, wherein the student can pay attention to different spots (the lecturer, whiteboard, slide presentation, the textbook, or another screen), the multi-video object, which contains several navigation controls besides the timeline (Figure 6.3(a)), allows the student to choose whether he wants to see more than one video at the same time (Figure 6.3(b)), or which video stream he wishes to see in full screen (Figure 6.3(c)).

Finally, the student has the facility to make annotations in the multimedia object by means of the watch-and-comment paradigm. For instance, he can mark some part of the lecture as important or irrelevant, or he can delimit a snippet of the lecture which he did not understand for further research or to ask the professor or tutor. He can also make comments on the lecture via audio or text, in similar in-classroom students do with paper and pencil.

6.4.4 Instrumented Classroom

The capture-tool prototype was deployed in a multi-purpose room (Figure 6.4). At the front of the room (Figure 6.4(a)) there is a conventional whiteboard, an electronic whiteboard and a notebook in which the presenter can browse the Web or use other software. The interactive whiteboard can be used to present slides (there is a Bluetooth presenter to control the presentation) and it allows drawing and writing over the screen. At the back of the room (Figure 6.4(b)) we placed two AVCHD, one with focus on the interactive whiteboard and the other with focus on the conventional whiteboard. We placed a webcam as a wide-shot cam, framing the whole front of the room. The cameras are locked cabinets when not in use.

We invited six instructors to use the prototype and record presentations. Four instructors recorded a lecture simulation (without students), one professor recorded a conventional lecture (with students), and one instructor recorded a problem solving class. We also used the prototype to record the presentation of term paper.

In the next sections, we report on results from analysing the interactions students had with the multimedia learning object resulting from the capture of the problem solving class.

6.5 Case Study: Capture Lecture

Using the capture-tool prototype, one instructor captured one lecture: the capture was made in several modules, without students in the classroom. The students had access to the multimedia learning object to prepare to their final exam.

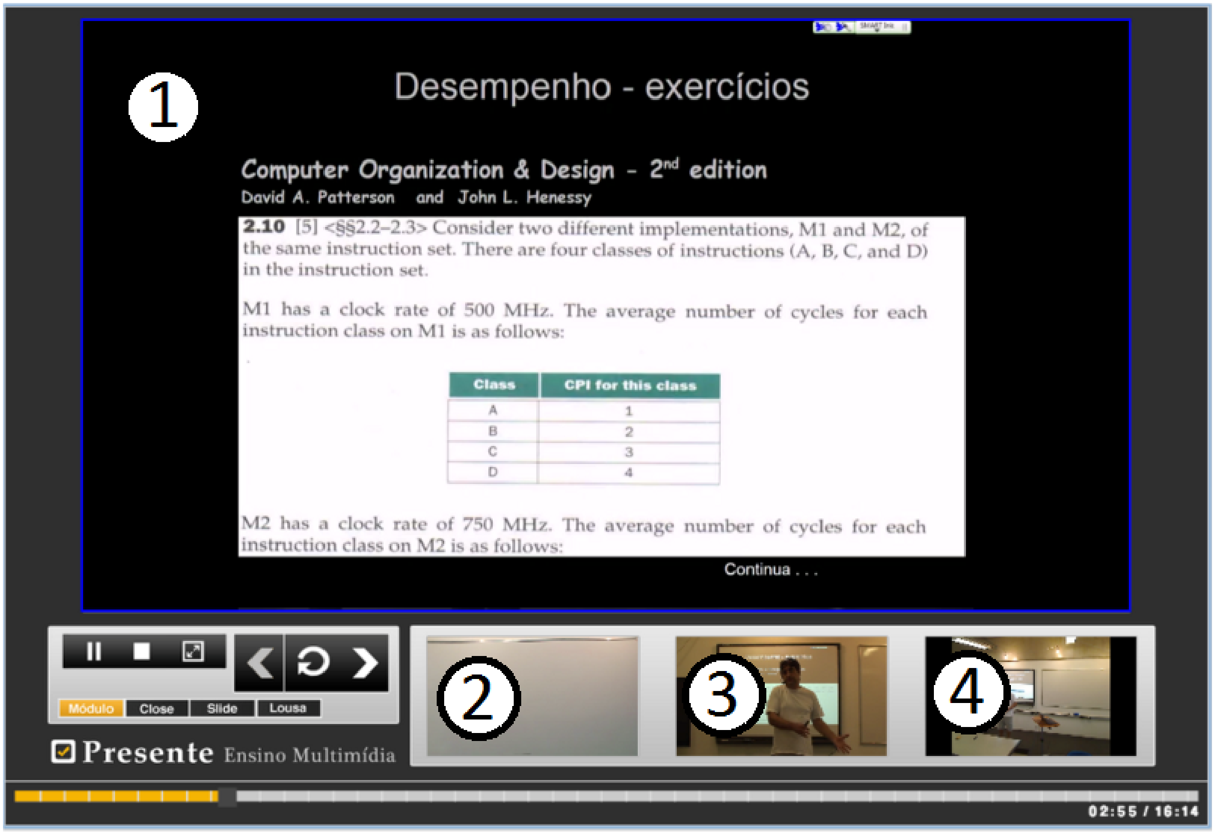
The lecture captured was a problem solving session for a Computer Organization course in which an instructor solved a total of 15 exercises. These exercises were related to each other and usually a subsequent exercise used some results from the previous one. The exercises also become more difficult as the presentation progressed.

The presentation was organized into 12 modules, performing a total of 1 hour and 18 minutes of content. The first 3 exercises were grouped in the module 1, module 5 contained 2 exercises, and all the other modules presented one exercise each.

Figure 6.5 depicts the multimedia object generated from the presentation. There are four streams: (1) the capture of the projected slide, which contained the description of the exercise; (2) the camera focused on the conventional whiteboard; (3) the camera focused on the slide; and (4) the wide-shot camera. Although the generation process has a feature that allows the automatic orchestration of the cameras (e.g., the automatic selection of which video stream would be presented in the main (bigger) window), in this study case we did not use it. The aim was to exploit the students' interaction, forcing them to choose, for a better learning experience, which would be the video to be presented in the main window at each instant.

The multimedia object was made available for the students in the Web and, using the WebNCL's log API, we logged all the interactions carried out by the students, such as when and where the users clicked and to which point they seek in the presentation timeline. The logged data were stored in a NoSQL database. We developed python scripts to extract information relative to how the students interacted with the multimedia object.

Figure 6.6 presents information about the time spent by the students, as well as the number of interactions they performed with the multimedia object. Each point in the horizontal axis



Desempenho - exercícios

Computer Organization & Design - 2nd edition

David A. Patterson and John L. Henessy

2.10 [5] <§§2.2–2.3> Consider two different implementations, M1 and M2, of the same instruction set. There are four classes of instructions (A, B, C, and D) in the instruction set.

M1 has a clock rate of 500 MHz. The average number of cycles for each instruction class on M1 is as follows:

Class	CPI for this class
A	1
B	2
C	3
D	4

M2 has a clock rate of 750 MHz. The average number of cycles for each instruction class on M2 is as follows:

Continua . . .

Figura 6.5: Problem Solving Presentation

represents a student (identified in the chart as letters from A to R). The blue bars show the amount of time each student spent watching the multimedia object (left vertical axis) and red bars show the number of interactions each student performed (right vertical axis).

The total duration of the 12 modules was 1 hour and 18 minutes. Eighteen students watched the presentation for at least 4 minutes. The average playback time of these 18 students is 3542.67 seconds (about 59 minutes) with a standard deviation of 2382.23 seconds (about 39 minutes). The average number of interactions of the students is 118.55 with a standard deviation of 99.58.

Figure 6.7 summarizes the number of interactions of each category performed by the students. The interactions were organized in the following categories:

- Main Video Selections: interactions carried out by the students in order to change the main video stream;
- Play/Pause: interactions causing the pause and the resume of the playblack;
- Timeline navigation: interactions that cause a move forward or backward through the timeline;
- Module Navigation: interactions that cause the change of the module currently watched;

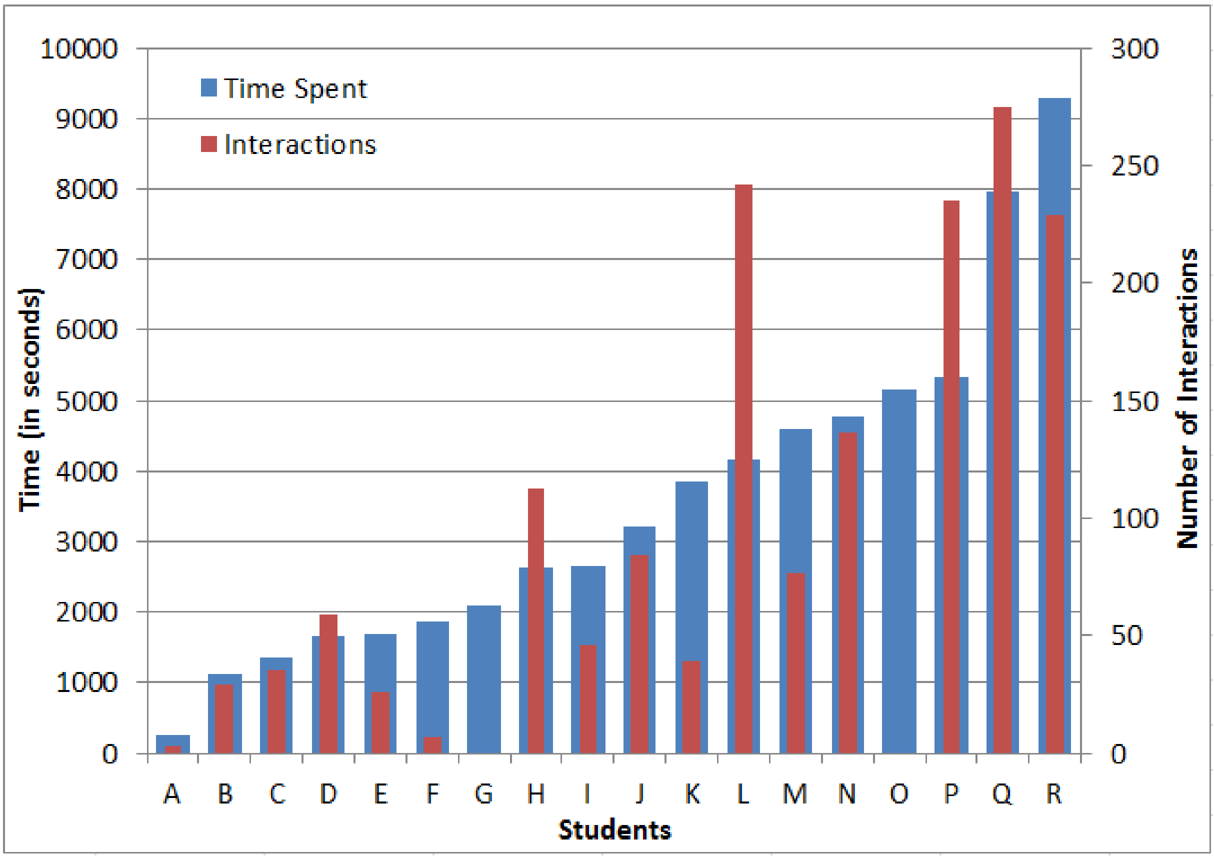


Figura 6.6: Students Interactions

- Points of Interest: interactions resulting from navigation by points of interest (e.g. slide transitions).

Figure 6.8 summarizes how much time each module was watched. In order to get a better visualization, the values in the left vertical axis were normalized by the module time length. The blue bars represent the time in which the presentation was running (not paused) and the red bars are the time in which the presentation was paused. The green line represents the number of students that watched each module for at least 10% of their time length. The figure suggests that the modules in which the students spent more time were the module 2 and module 4. It also suggests that the number of different students that watched the modules decreases as the presentation progress

Figure 6.9 summarizes the watching attendance of some modules. The horizontal axis is the number of seconds of each module (Presentation Space). The blue line represents the number of times the instant was watched by students, and the red line the number of different students that watched each instant.

As the modules always start from second 0, it is natural that the attendance of the first seconds is bigger. The points where the blue line is above the red line mean that the moment

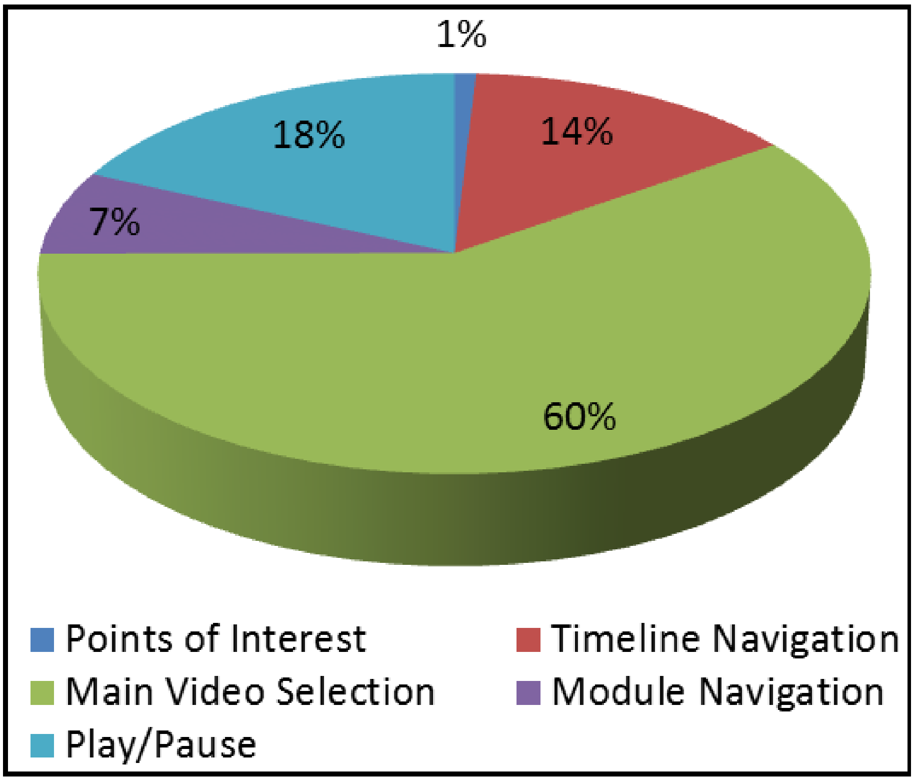


Figura 6.7: Interactions per Category

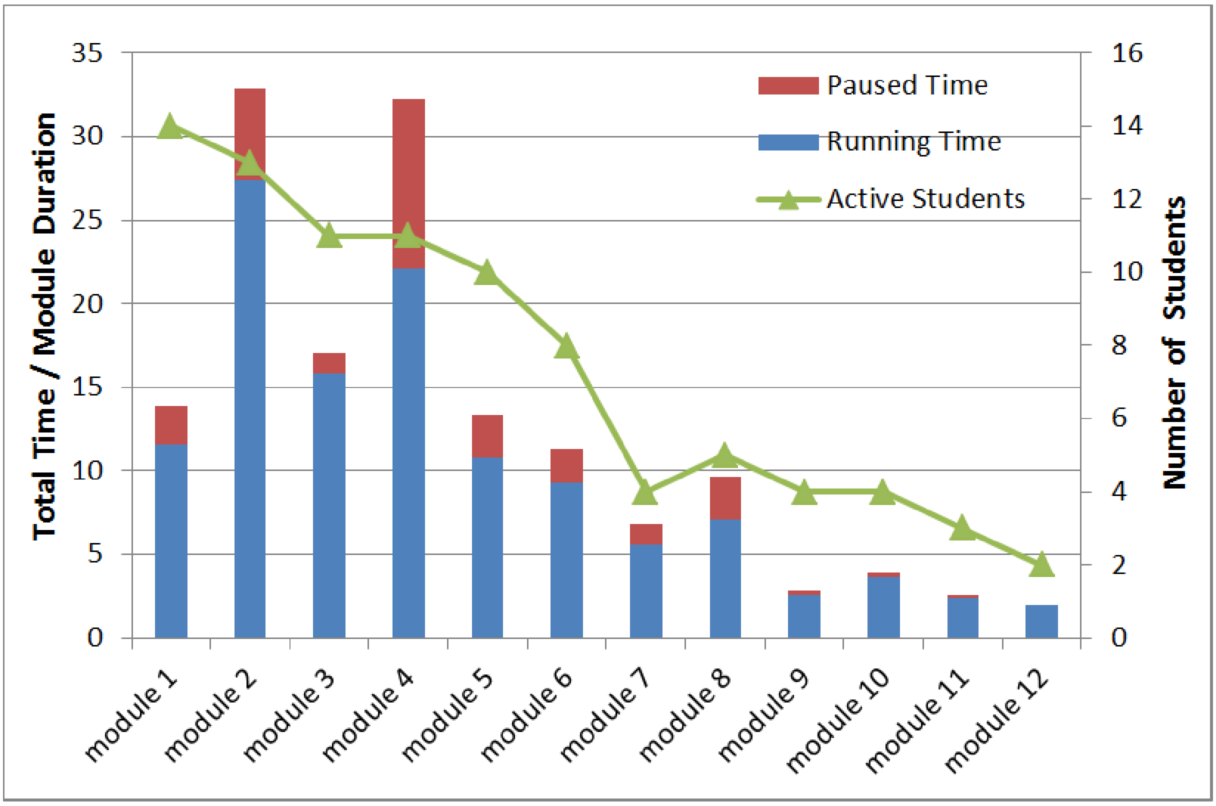
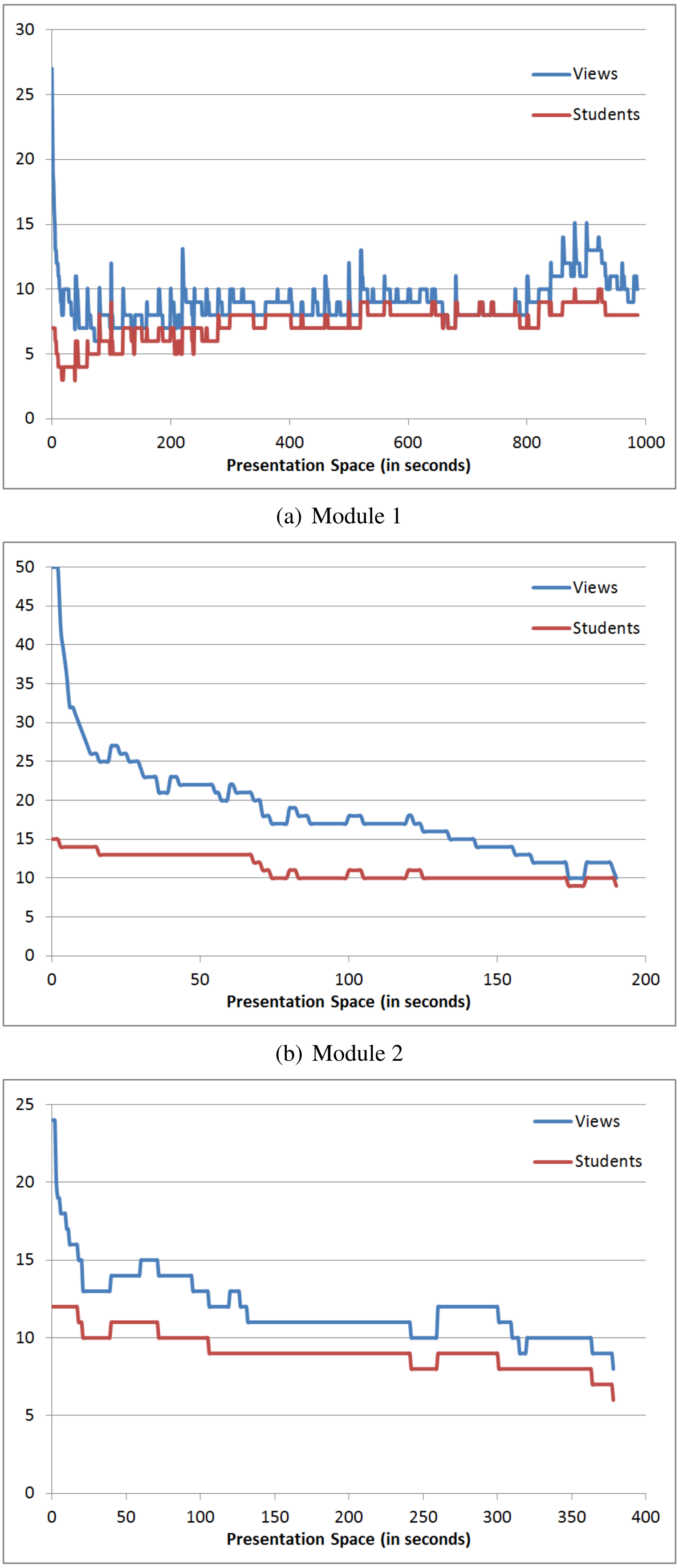


Figura 6.8: Presentation Modules Statistics



(a) Module 1

(b) Module 2

(c) Module 4

Figura 6.9: Modules Attendance

was watched more than once by the same students. This graphic can be useful for lecturers to find out which parts of a lecture are more useful or important for the students, or even to identify points where students have difficult to understand. For instance, after the second 800 in Module 1 (Figure 6.9(a)), the blue line deviates from red line, it suggests that that segment of module 1 were watched more times by the students.

Given that the multimedia object has more than one video stream and that the students can choose which stream they wish to see as the main stream, the information of which stream is most selected as the main stream at each moment can be useful.

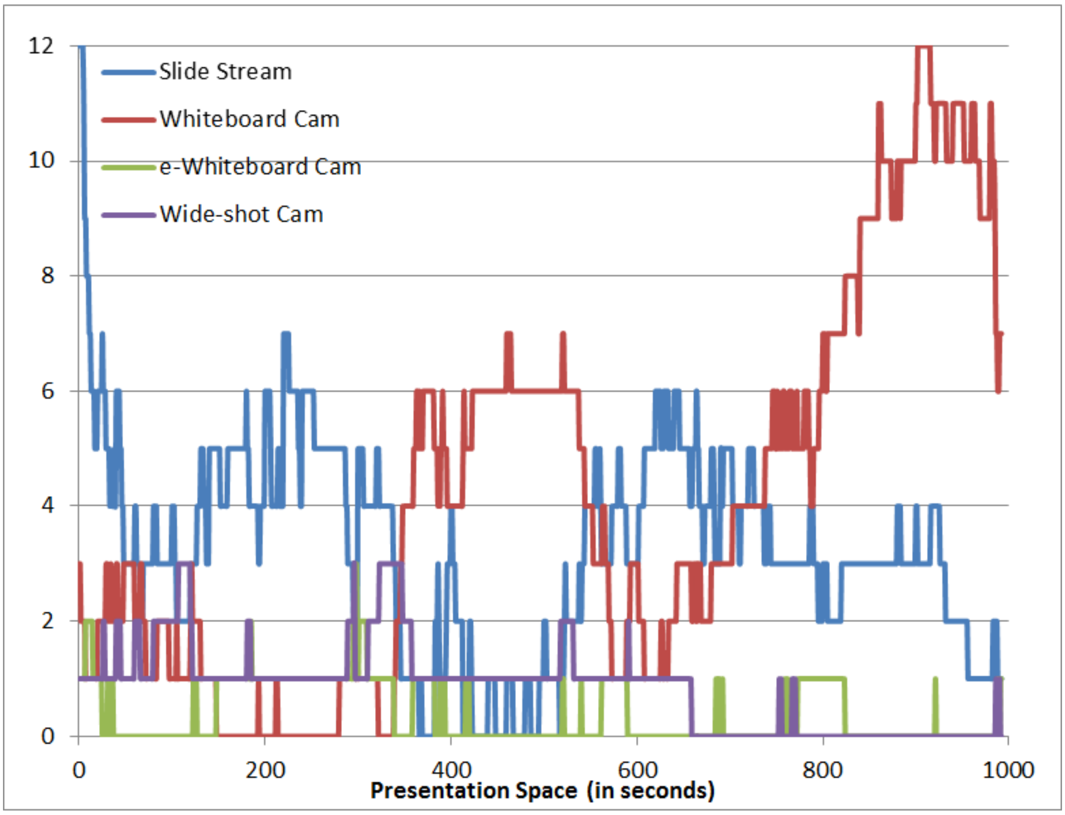
Figure 6.10(a) and Figure 6.10(b) summarize which streams were most selected as the main stream in each moment of, respectively, module 1 and module 4. Each line represents how many times a stream was watched in a specific moment. The blue line refers to the slide projection capture (Figure 6.5(1)); the red refers to the camera focused on the conventional whiteboard (Figure 6.5(2)); the green camera focused on the slide presentation (Figure 6.5(3)); and the purple the wide-shot camera (Figure 6.5(4)).

According to Figure 6.10(a), the more watched streams were the slide presentation and the whiteboard camera. We can also note that the slide presentation is more watched near the moments when there are slide transitions in the module 1 (seconds 213 and 515). Figure 6.10(b) suggests that after the second 100 the predominant stream was the whiteboard camera stream.

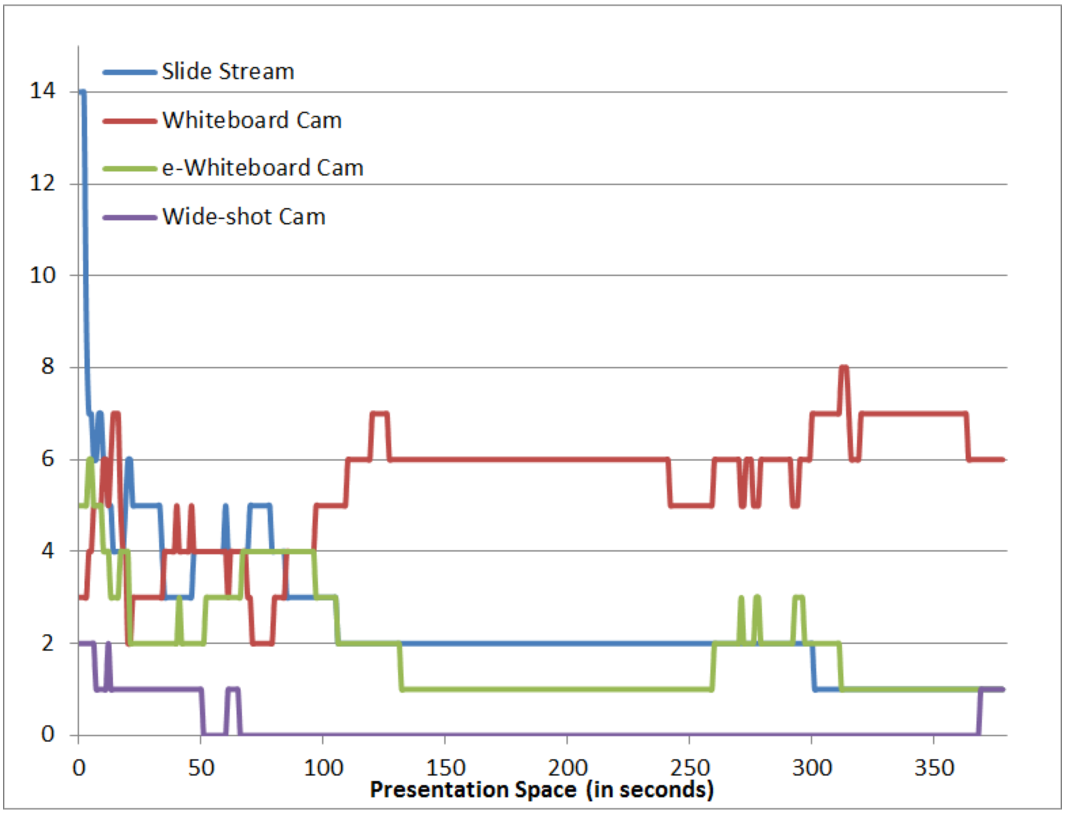
Figure 6.11 illustrates the behavior of 2 students when interacting with the presentation for module 1, 2 and 4. The student P (blue line) and Q (red line) are the same students from the Figure 6.6. The horizontal axis is the playback timeline and the vertical axis is the presentation timeline (presentation space). Vertical straight lines represent a navigation that the student performed during playback and horizontal straight lines represents moments in which the student paused the presentation. These graphics allow to visualize how a student interact with the presentation in detail. For instance, we can observe in Figure 6.11(a) that student P starts watching from second 180 and performed some backward moves mainly in the end of the presentation. Student Q watched almost linearly until second 650 and then returned to the beginning of the presentation and watched it again until the end performing some pauses.

6.6 Lessons Learned

The graphics were presented to the instructor. He analysed them taking into account the content of his presentation, how it was presented and which and how students interacted with it.

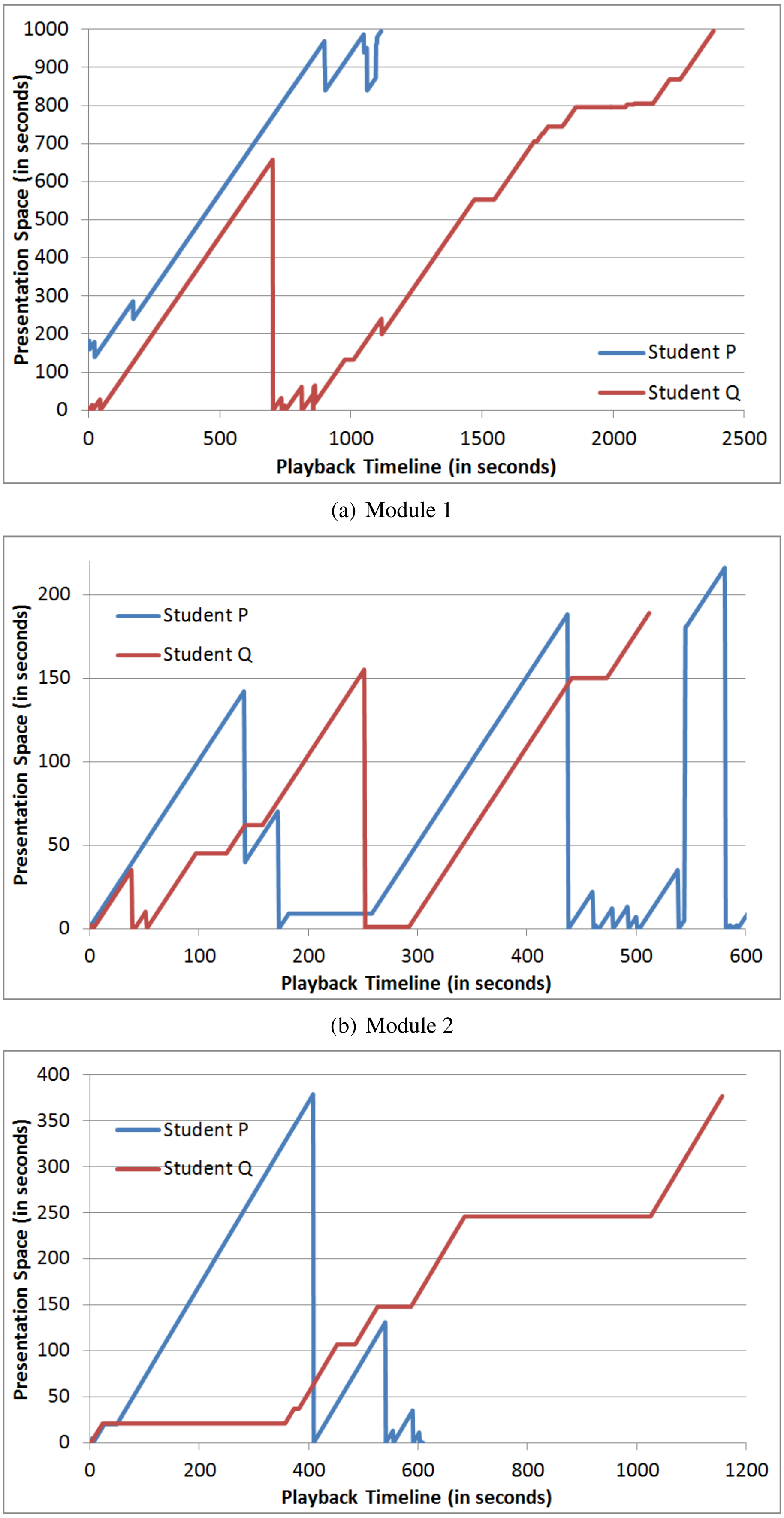


(a) Module 1



(b) Module 4

Figura 6.10: Streams View



(a) Module 1

(b) Module 2

(c) Module 4

Figura 6.11: Students Navigation

His first observation: “the graphics are very abstract for a teacher to analyse them by himself”. As as consequence, the remaining analysis was carried out then with the help of one of the authors. From now on, what is reported in this section is a combination of what the teacher observed and some conclusions of the authors.

Not all students interacted with the multimedia learning object, even knowing that it could have tips for the exam. Reasons for this may have been the commitments of students with other exams, the late release of the learning object (two days before the exam), and also its long duration, about one hour and twenty minutes (4800 s). As shown in Figure 6.8, several of the students only watched the first modules. Besides the reasons already mentioned, some of them may have found the presentation boring. A questionnaire with explicit questions could help understand this attitude.

Students were able to view the slides presented in two ways, watching the video of the instructor presenting (and maybe interacting with) the slide on the interactive whiteboard, or watching the slide captured directly from the output of the projector (best quality). The preference was for the latter, as shown in Figure 6.10(a) with the blue and green lines. It is likely that the type of the presentation, without many interactions with the interactive whiteboard, does not justify the view of the slide in lower quality.

The resulting multimedia learning object may consider context information. It can then guarantee that the focus of the presentation, at every moment, be automatically taken to the main window display. So, when the teacher uses the whiteboard, her or his video could be automatically selected to the main window. The same applies for the videos associated with the interactive whiteboard, the application captures, etc. However, we chose to force the student himself to perform all the video switching. Some students expressed frustration with such duty. The goal was to keep them alert to the presentation in order to make it less monotonous. The strategy worked. As shown in Figure 6.7, 60% of the interactions (975) were used for selecting the video to the main window. The effectiveness of the strategy in terms of learning, however, needs to be evaluated.

Figure 6.7 also shows the limited use of the navigation using Points of Interest. Students preferred to use the Timeline (14% of interactions) to control the presentation. Two reasons may be related to this: students are used to the paradigm of watching video in the Web; and the lesson has not encouraged or justified the need for this type of navigation. However, the navigation through the modules happened with a frequency (7%) corresponding to the one expected (and planned) by the teacher.

Figure 6.9(a) shows that an almost constant public watched module 1 (in terms of number of student). However, the blue line shows some peaks in visits to some parts of the presentation, in terms of times the segment was played. The moments around the 900th second are the evident ones. The analysis of the video in those moments, carried out by the teacher, indicates that the subject could be presented more clearly – that is, there is room for improvement in the way the presentation was made.

Modules 2 and 4 were the most popular, not the 1 as expected for being the first. The visiting time was normalized by the duration of the module in Figure 6.8. As the first module was the one with the longer duration, it may indicate that large modules are more verbose (which was confirmed by the teacher for module 1) and therefore tend to be somewhat repetitive. Moreover, this feature can be further studied since the content of module 1 was less complex than the others.

The navigation patterns, illustrated in Figure 6.6, show different behaviors by the students. There are students who simply "watch" the presentation and do not perform any interaction at all, even to change the video in the main window, as was the case of the students G and O. These, probably thinking they would be evaluated by their performing in viewing the presentation, let the presentation run without perhaps give attention to it. Others, such as students L and P, watched all the presentation and performed many interactions. There are also students, as Q and R, who, besides interacting a lot, also watched repeatedly parts of the presentation, nearly doubling the original time of the presentation. The figures show that the number of interactions was proportional to the time in which the presentation was watched, which indicates a similar degree of interactivity between the students in the class. Another interesting observation about the behavior of students was made by the teacher: "one of the students who watched and interacted the most with the multimedia learning object, the student N, usually shows a very apathetic behavior in the classroom". This may indicate that interactive multimedia learning objects, generated by capturing multimodal and multi-device presentations, may be a good option for students who like to be in control of what they pay attention to.

6.7 Final Remarks

Extra-class material may be offered to students in the form of multimedia objects that integrates synchronized text, image, audio and video explanations on the studied subject. A learning object like this can be produced in studios, with support of various professionals. Alternatively, as is the case presented in this paper, the multimedia object can be automatically generated from the ubiquitous capture of a traditional lecture in the classroom. The lecture can be delivered to a group of students, or be delivered to an empty classroom just for capture purposes. Context information informing moments of interest such as slide transitions can be included in the multimedia object to provide students with semantic navigation.

The multimedia object should be instrumented to log the navigation performed by students so that, besides acting as extra-class material, they can be effective as tool that provides feedback which contributes to improve its own content. In the situation presented in this paper, it is the instructor who receives the feedback, which she can analyse to identify improvements not only in terms of the content itself but also in terms of how the exposition was made at the time of capture.

The case study presented suggests how similar analyses that can be performed in other presentations, even though only a portion of the logged information was used. As a result, the analysis is useful both as a reference for the preparation of presentations used in research involving interactive multimedia objects, and in the research in Education.

Regarding future work, we plan to investigate alternatives for: (a) the enrichment of the graphic interface of the multimedia object so as to improve interactivity; (b) the capture of more contextual information during the presentation toward providing novel navigation facilities; (c) the development of visualization tools for the instructor to analyse the information captured while the students interacted with the multimedia object. The aim is to built a general infrastructure that helps building similar capture-based applications (PIMENTEL; JR.; CATTELAN, 2007).

We also to conduct interdisciplinary research toward better understanding the impact, on education, of the use of multimedia learning objects built from the capture of multimodal and multi-device presentations.

The teacher also noted a relationship between student's performance on assessment on the subject of the presentation and the time each one spent with the multimedia learning object. Most who watched and interacted with all modules of the presentation performed well. The individual analysis of each student can be performed using graphs similar to those shown in Figure 6.11, for instance.

Capítulo 7

CONCLUSÃO

7.1 Contribuições e Limitações

Este trabalho apresenta uma infraestrutura para a captura de apresentações educacionais. Os processos de aperfeiçoamento do software para captura e da sala instrumentada, apoiados em estudos de casos realizados com professores e alunos apresentam contribuições para a computação ubíqua:

- O modelo apresentado na Seção 2.3 pode servir como referência para o desenvolvimento de outros sistemas para captura & acesso de atividades humanas;
- Foram identificados pontos de interesse que ajudam a reconstituir o contexto das apresentações educacionais capturadas; e
- A utilização do paradigma de aulas presenciais como base para o processo de captura torna a produção de conteúdo multimídia simples e natural para professores.

Entretanto, o protótipo desenvolvido não é capaz de identificar todos os tipos de pontos de interesse listados neste trabalho. Além disso, a divisão explícita das apresentações em módulos torna o processo de captura mais distante de uma aula presencial convencional. Uma possibilidade a ser explorada seria utilizar uma combinação de fatores para se tentar realizar a quebra em módulos de forma automática, como momentos de silêncio, palavras-chave ou marcações específicas em slides.

Apesar do modelo de captura & acesso apresentado na seção 2.3 ser genérico e, teoricamente, poder ser utilizado para capturar outros tipos de apresentações educacionais, o protótipo desenvolvido foca em apresentações que se assemelham a aulas presenciais expositivas. Dessa forma, não foi possível validar a generalidade do modelo.

Do posto de vista de sistemas multimídia e Web, sobretudo para a comunidade de TV Digital Interativa, WebNCL representa uma contribuição prática importante. Além de permitir a reprodução de vídeo interativo em uma grande variedade de dispositivos, WebNCL é uma ferramenta fácil de ser integrada ou estendida, o que colabora com a pesquisa em mídias interativas e TV Digital. Como limitações, WebNCL ainda não suporta totalmente a linguagem NCL 3.0. Além disso, a não homogeneidade dos navegadores também impõe requisitos que precisam estar sendo considerados para garantir a universalidade de WebNCL (Seção 3.6).

Ainda na área de sistemas multimídia, os componentes desenvolvidos para a geração dos objetos de aprendizagem cobrem funcionalidades comuns — porém não triviais de serem representadas em linguagens declarativas — a muitas classes de apresentações multimídia. Dessa forma, o reuso dos componentes, ou mesmo das estratégias utilizadas para a geração de código, podem facilitar a criação de apresentações multimídia complexas e/ou altamente interativas, principalmente para profissionais não familiarizados com paradigmas de programação de mais baixo nível.

As técnicas para permitir o controle remoto de múltiplas instâncias de apresentações multimídia contribuem para TV Social e outras áreas interessadas no consumo de mídia distribuída. Em menor grau, as técnicas apresentam contribuições para a área de sistemas distribuídos. Entretanto, é importante ressaltar que o protótipo para pilotagem remota de apresentações não considera a sincronização de mídia distribuída. O modelo de pilotagem remota também não se adequa para aplicações em que múltiplos usuários precisam compartilhar suas interações simultaneamente, como acontece em jogos online.

Os estudos de caso realizados, bem como o ferramental desenvolvido para capturar e extrair dados dos logs de interação, podem levar a contribuições na área de Interação Humano Computador. Porém, mais estudos de caso, com variáveis controladas, devem ser elaborados a fim de se confirmar os indícios sobre como os estudantes interagem com os objetos de aprendizagem multimídia e como isso pode fornecer pistas aos professores de como melhorar suas apresentações educacionais (Seções 6.5 e 6.6).

Muito do esforço realizado durante o programa de mestrado foi concentrado na implementação dos protótipos necessários para a condução desta pesquisa. Uma vez que os protótipos estão prontos e funcionais, estudos de caso específicos podem ser realizados, explorando diferentes cenários com alunos e professores. Outras contribuições na área de Interação Humanam-Computador deverão decorrer dos estudos de caso realizados com os protótipos desenvolvidos.

O trabalho também deixa como legado um ferramental para a pesquisa na área de educação e informática na educação. As implicações da utilização de objetos de aprendizagem multimídia devem ser investigadas com a colaboração de pesquisadores da área da educação.

7.2 Lições Aprendidas

No desenvolvimento da WebNCL, percebeu-se que a forma como os navegadores implementam a orientação a eventos de JavaScript facilita o desenvolvimento de máquinas de apresentação multimídia baseadas em relações de casualidade, como é o caso do NCL.

No primeiro contato entre alunos em uma sitauação real e os objetos de aprendizagem multimídia, a interface do objeto não oferecia a opção de navegação pela linha do tempo. Essa ausência frustrou os alunos, que estão habituados a navegar por vídeos utilizando esse paradigma.

Os professores não estão habituados a utilizar todos os recursos oferecidos pela sala instrumentada. Recomendações e boas práticas para tirar maior proveito dessas facilidades pode favorecer a geração de objetos de aprendizagem multimídia mais interessantes.

A implementação da navegação por linha do tempo em linguagens declarativas, assim como outras funcionalidades descritas na Seção 4.3, pode ser complexa. Isso sugere que linguagens declarativas para sincronização podem ser aperfeiçoadas para permitir formas de interação mais ricas, comuns na Web ou em dispositivos móveis.

A utilização de frameworks no desenvolvimento dos protótipos, como o popcorn.js e jquery no WebNCL, e o QT [1] na ferramenta de captura, facilitou na portabilidade dos protótipos. A ferramenta de captura havia sido concebida para executar em ambiente Unix, porém devido a problemas de compatibilidade do ambiente Unix com a lousa eletrônica, foi necessário portar a ferramenta para ambiente Windows. O porte da ferramenta foi simples, exigindo modificações em apenas algumas linhas de código.

A separação de processos complexos em etapas independentes facilita na recuperação de erros. Isso pode ser observado na funcionalidade de separar a captura das apresentações educacionais em módulos. Além disso, o processo de geração dos objetos multimídia, por ser composto de atividades com alto custo computacional como técnicas de processamento de imagem e conversão de vídeo, também foi estruturado em etapas independentes. Dessa forma, se algum erro acontecer em uma etapa, o processo não precisa ser refeito do inicio, e sim da última etapa realizada com sucesso.

A análise da interação dos alunos com objetos de aprendizagem pode oferecer muitos indí-

[1] Qt Project – http://qt-project.org/

cios interessantes para o professor melhorar suas apresentações multimídia. Além disso, pode-se extrair pontos de interesse de uma aula ao sumarizar os segmentos de aula mais visitados pelos alunos.

As lições aprendidas sobre a forma como os alunos interagem com os objetos multimídia estão sumarizadas na Seção 6.6.

7.3 Trabalhos Futuros

Captura & Acesso de apresentações educacionais é um tópico de pesquisa recorrente. Entretanto, esse tópico oferece várias oportunidades de pesquisa que ainda não foram exploradas:

- Identificação e exploração de novos pontos de interesse, sobretudo para outros tipos de atividades menos centradas no apresentador;
- Exploração de ferramentas portáveis, capazes de capturar apresentações realizadas em computadores pessoais ou tablets;
- Adaptação dos objetos de aprendizagem multimídia para outros dispositivos ou para diferentes públicos;
- Captura de outros tipos de atividades humanas ligadas à educação, como debates ou atividades em grupos;
- Explorar como os pontos de interesse e outras funcionalidades oferecidas pelo objeto de aprendizagem multimídia impactam na forma como os usuários interagem;
- Utilizar as próprias interações realizadas pelos estudantes para encontrar pontos de interesse, através de mineiração semântica;
- Utilizar anotações realizadas pelos estudantes (como perguntas) para inferir pontos de interesse;
- Utilizar sessões síncronas realizadas por professores, alunos e tutores para enriquecer os objetos educacionais num processo de aperfeiçoamento continuo;
- Questões de acessibilidade em relação aos objetos multimídia ou mesmo em relação à linguagens de sincronização declarativas;
- Estudo do impacto dos objetos de aprendizagem multimídia do ponto de vista pedagógico, em diferentes cenários como EaD, ensino presencial, etc.; e

- Definição de um framework para avaliar a sensação de presença, imerção e controle que um usuário tem ao interagir com um objeto de aprendizagem multimídia gerado a partir da captura de apresentações educacionais.

7.4 Considerações Finais

MOOC e EaD estão se tornando cada vez mais populares. Apesar das várias vantagens que tais modelos de ensino-aprendizagem trazem, como a maior autonomia dos estudantes ou flexibilidade dos horários; a ausência do contato face-a-face pode criar um distanciamento entre professores e alunos, o que pode influenciar negativamente no aprendizado ou causar desistência por parte dos alunos.

A utilização de TICs pode colaborar na redução dessa distância. Alunos podem interagir com objetos de aprendizagem que tentam reproduzir a experiência de aulas presenciais, além de oferecer outras facilidades apenas possíveis em meios mediados por computador. TICs que anteriormente viabilizaram o conceito de educação a distância, agora podem permitir que estudantes geograficamente espalhados sintam-se “presentes” em uma sala de aula.

Referências

ABED. *CensoEAD.BR. Relatório Analítico da Aprendizagem a Distância no Brasil*. [S.l.], 2009.

ABNT, N. *Associação Brasileira de Normas Técnicas. 2007. Digital Terrestrial Television Standard 06: Data Codification and Transmission Specifications for Digital Broadcasting*. [S.l.], 2007.

ABOWD, G. et al. Anchoring discussions in lecture: an approach to collaboratively extending classroom digital media. In: *Proceedings of the 1999 conference on Computer support for collaborative learning*. International Society of the Learning Sciences, 1999. (CSCL '99). Disponível em: <http://dl.acm.org/citation.cfm?id=1150240.1150241>.

ABOWD, G. D. et al. Teaching and learning as multimedia authoring: the classroom 2000 project. In: *Proceedings of the fourth ACM International Conference on Multimedia*. New York, NY, USA: ACM, 1996. (MULTIMEDIA '96), p. 187–198. ISBN 0-89791-871-1. Disponível em: <http://doi.acm.org/10.1145/244130.244191>.

AZEVEDO, R. G. D. A.; SOARES, L. F. G. Embedding 3d objects into ncl multimedia presentations. In: *Proceedings of the 17th International Conference on 3D Web Technology*. New York, NY, USA: ACM, 2012. (Web3D '12), p. 143–151. ISBN 978-1-4503-1432-9. Disponível em: <http://doi.acm.org/10.1145/2338714.2338739>.

BAUERLEIN, M. *The Chronicle of Higher Education - Lectures On the Benefits*. 2011. Disponível em: <Http://chronicle.com/blogs/brainstorm/on-the-benefits-of-lectures/36519>.

BECK, K.; ANDRES, C. *Extreme programming explained: embrace change*. [S.l.]: Addison-Wesley Professional, 2004.

BIANCHI, M. Automatic video production of lectures using an intelligent and aware environment. In: *Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia*. New York, NY, USA: ACM, 2004. (MUM '04), p. 117–123. ISBN 1-58113-981-0. Disponível em: <http://doi.acm.org/10.1145/1052380.1052397>.

BOLL, S. *MM4U - A framework for creating personalized multimedia content*. 2003.

BONWELL, C. C. Enhancing the lecture: Revitalizing a traditional format. *New Directions for Teaching and Learning*, 1996. v. 1996, p. 31–44, 1996.

BORONAT, F.; LLORET, J.; GARCíA, M. Multimedia group and inter-stream synchronization techniques: A comparative study. *Information Systems*, 2009. v. 34, n. 1, p. 108 – 131, 2009. ISSN 0306-4379. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0306437908000525>.

BORONAT, F. et al. The need for inter-destination synchronization for emerging social interactive multimedia applications. In: . [S.l.: s.n.], 2012. p. 150–158.

BOUYAKOUB, S.; BELKHIR, A. Smil builder: An incremental authoring tool for smil documents. *ACM Trans. Multimedia Comput. Commun. Appl.*, 2011. ACM, New York, NY, USA, v. 7, n. 1, p. 2:1–2:30, fev. 2011. ISSN 1551-6857. Disponível em: <http://doi.acm.org/10.1145/1870121.1870123>.

BRADSKI, G. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000. 2000.

BROTHERTON, J. A.; ABOWD, G. D. Lessons learned from eclass: Assessing automated capture and access in the classroom. *ACM Trans. Comput.-Hum. Interact.*, 2004. ACM, New York, NY, USA, v. 11, n. 2, p. 121–155, jun. 2004. ISSN 1073-0516. Disponível em: <http://doi.acm.org/10.1145/1005361.1005362>.

BULTERMAN, D. C.; RUTLEDGE, L. W. *SMIL 3.0: Flexible Multimedia for Web, Mobile Devices and Daisy Talking Books*. 2nd ed.. ed. [S.l.]: Springer Publishing Company, Incorporated, 2008. ISBN 3540785469, 9783540785460.

CASHIN, W. E. idea paper no. 14 - improving lectures. 1985. "September"1985.

CATTELAN, R. G.; BALDOCHI, L. A.; PIMENTEL, M. D. G. Experiences on building capture and access applications. In: *In Proceedings of the 9th Brazilian Symposium on Multimedia and Hypermedia Systems*. [S.l.: s.n.], 2003. p. 112–127.

CATTELAN, R. G. et al. Watch-and-comment as a paradigm toward ubiquitous interactive video editing. *ACM Trans. Multimedia Comput. Commun. Appl.*, 2008. ACM, New York, NY, USA, v. 4, n. 4, p. 28:1–28:24, nov. 2008. ISSN 1551-6857. Disponível em: <http://doi.acm.org/10.1145/1412196.1412201>.

CAZENAVE, F.; QUINT, V.; ROISIN, C. Timesheets. js: Tools for web multimedia. In: ACM. *Proceedings of the 19th ACM international conference on Multimedia*. [S.l.], 2011. p. 699–702.

CAZENAVE, F.; QUINT, V.; ROISIN, C. Timesheets.js: when smil meets html5 and css3. In: *Proceedings of the 11th ACM symposium on Document engineering*. New York, NY, USA: ACM, 2011. (DocEng '11), p. 43–52. ISBN 978-1-4503-0863-2. Disponível em: <http://doi.acm.org/10.1145/2034691.2034700>.

CESAR, P.; BULTERMAN, D.; JANSEN, A. The ambulant annotator: empowering viewer-side enrichment of multimedia content. In: ACM. *Proceedings of the 2006 ACM symposium on Document engineering*. [S.l.], 2006. p. 186–187.

CHARLAND, A.; LEROUX, B. Mobile application development: web vs. native. *Communications of the ACM*, 2011. ACM, v. 54, n. 5, p. 49–53, 2011.

CHOU, H.-P. et al. Automated lecture recording system. In: *System Science and Engineering (ICSSE), 2010 International Conference on*. [S.l.: s.n.], 2010. p. 167 –172.

CORALLO, A.; CAPUTO, E.; CISTERNINO, V. Business modelling language: a framework supporting interoperability in cluster of smes. In: *Digital EcoSystems and Technologies Conference, 2007. DEST '07. Inaugural IEEE-IES*. [S.l.: s.n.], 2007. p. 107 –112.

COSTA, R. M. de R. et al. Live editing of hypermedia documents. In: *Proceedings of the 2006 ACM symposium on Document engineering*. New York, NY, USA: ACM, 2006. (DocEng '06), p. 165–172. ISBN 1-59593-515-0. Disponível em: <http://doi.acm.org/10.1145/1166160.1166202>.

CRINON, R. The dsm-cc object carousel for broadcast data services. In: IEEE. *Consumer Electronics, 1997. Digest of Technical Papers. ICCE., International Conference on*. [S.l.], 1997. p. 246–247.

CRUZ, V.; MORENO, M.; SOARES, L. Ginga-ncl: implementaçao de referência para dispositivos portáteis. In: ACM. *Proceedings of the 14th Brazilian Symposium on Multimedia and the Web*. [S.l.], 2008. p. 67–74.

DAMASCENO, J.; SANTOS, J. dos; MUCHALUAT-SAADE, D. Editec: hypermedia composite template graphical editor for interactive tv authoring. In: *Proceedings of the 11th ACM symposium on Document engineering*. New York, NY, USA: ACM, 2011. (DocEng '11), p. 77–80. ISBN 978-1-4503-0863-2. Disponível em: <http://doi.acm.org/10.1145/2034691.2034708>.

DICKSON, P. E. et al. Evaluation of automatic classroom capture for computer science education. In: *Proceedings of the fifteenth annual conference on Innovation and technology in computer science education*. New York, NY, USA: ACM, 2010. (ITiCSE '10), p. 88–92. ISBN 978-1-60558-820-9. Disponível em: <http://doi.acm.org/10.1145/1822090.1822116>.

DICKSON, P. E. et al. Student reactions to classroom lecture capture. In: *Proceedings of the 17th ACM annual conference on Innovation and technology in computer science education*. New York, NY, USA: ACM, 2012. (ITiCSE '12), p. 144–149. ISBN 978-1-4503-1246-2. Disponível em: <http://doi.acm.org/10.1145/2325296.2325334>.

ECMASCRIPT, E.; ASSOCIATION, E. C. M. et al. *ECMAScript Language Specification*.

ETSI, T. 102 819 v1. 3.1 (2005) - digital video broadcasting (dvb): Globally executable mhp (gem) specification 1.0. 2. *European Telecommunications Standards Institute, TS*. v. 102, n. 819, p. V1.

FERREIRA, G. et al. Ginga-ncl em dispositivos portáteis: Uma implementação para a plataforma android. In: *Proc. Brazilian Symposium on Multimedia and Web (WebMedia)*. [S.l.: s.n.], 2010.

FLEURY, C. et al. Architectures and mechanisms to efficiently maintain consistency in collaborative virtual environments. *Proc. of Software Engineering and Architectures for Realtime Interactive Systems-SEARIS*, 2010. p. 87–94, 2010.

FREITAS, G. D.; TEIXEIRA, C. Ubiquitous services in home networks offered through digital tv. In: ACM. *Proceedings of the 2009 ACM symposium on Applied Computing*. [S.l.], 2009. p. 1834–1838.

GAGGI, O.; DANESE, L. A smil player for any web browser. In: *Proceedings of International Conference on Distributed Multimedia Systems*. Firenze, Italy: [s.n.], 2011. p. 114–119.

GAMMA, E. *Design patterns: elements of reusable object-oriented software*. [S.l.]: Addison-Wesley Professional, 1995.

GASPAR, T. C.; PRADO, A. F. do; TEIXEIRA, C. A. C. Software product lines for web 2.0 synchronous collaboration. In: *Proceedings of the XV Brazilian Symposium on Multimedia and the Web*. New York, NY, USA: ACM, 2009. (WebMedia '09), p. 11:1–11:8. ISBN 978-1-60558-880-3. Disponível em: <http://doi.acm.org/10.1145/1858477.1858488>.

GEERTS, D. et al. Are we in sync?: synchronization requirements for watching online video together. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. New York, NY, USA: ACM, 2011. (CHI '11), p. 311–314. ISBN 978-1-4503-0228-9. Disponível em: <http://doi.acm.org/10.1145/1978942.1978986>.

H.761, R. I.-T. *Nested Context Language (NCL) and Ginga-NCL for IPTV Services*. [S.l.], 2009.

HALAWA, S. et al. Classx: an open source interactive lecture streamingsystem. In: *Proceedings of the 19th ACM international conference on Multimedia*. New York, NY, USA: ACM, 2011. (MM '11), p. 719–722. ISBN 978-1-4503-0616-4. Disponível em: <http://doi.acm.org/10.1145/2072298.2072428>.

HüRST, W. Indexing, searching, and skimming of multimedia documents containing recorded lectures and live presentations. In: *Proceedings of 11th ACM International Conference on Multimedia*. New York, NY, USA: ACM, 2003. (MULTIMEDIA '03), p. 450–451. ISBN 1-58113-722-2. Disponível em: <http://doi.acm.org/10.1145/957013.957107>.

JANSEN, J.; BULTERMAN, D. Enabling adaptive time-based web applications with smil state. In: ACM. *Proceeding of the eighth ACM symposium on Document engineering*. [S.l.], 2008. p. 18–27.

JANSEN, J.; BULTERMAN, D. Smil state: an architecture and implementation for adaptive time-based web applications. *Multimedia Tools and Applications*, 2009. Springer, v. 43, n. 3, p. 203–224, 2009.

JANSEN, J.; CESAR, P.; BULTERMAN, D. C. A model for editing operations on active temporal multimedia documents. In: *Proceedings of the 10th ACM symposium on Document engineering*. New York, NY, USA: ACM, 2010. (DocEng '10), p. 87–96. ISBN 978-1-4503-0231-9. Disponível em: <http://doi.acm.org/10.1145/1860559.1860579>.

JAVADTV, A. *Java DTV API 1.3 Specification, Sun Microsystems (2009)*. 2010.

KEGEL, I. et al. Enabling 'togetherness' in high-quality domestic video. In: *Proceedings of the 20th ACM international conference on Multimedia*. New York, NY, USA: ACM, 2012. (MM '12), p. 159–168. ISBN 978-1-4503-1089-5. Disponível em: <http://doi.acm.org/10.1145/2393347.2393375>.

KING, P.; SCHMITZ, P.; THOMPSON, S. Behavioral reactivity and real time programming in xml: functional programming meets smil animation. In: *Proceedings of the 2004 ACM Symposium on Document Engineering*. New York, NY, USA: ACM, 2004. (DocEng '04), p. 57–66. ISBN 1-58113-938-1. Disponível em: <http://doi.acm.org/10.1145/1030397.1030411>.

LAMPI, F.; KOPF, S.; EFFELSBERG, W. Automatic lecture recording. In: *Proceedings of the 16th ACM international conference on Multimedia*. New York, NY, USA: ACM, 2008. (MM '08), p. 1103–1104. ISBN 978-1-60558-303-7. Disponível em: <http://doi.acm.org/10.1145/1459359.1459583>.

LANIR, J.; BOOTH, K. S.; TANG, A. Multipresenter: a presentation system for (very) large display surfaces. In: *Proceedings of the 16th ACM International Conference on Multimedia*. New York, NY, USA: ACM, 2008. (MULTIMEDIA '08), p. 519–528. ISBN 978-1-60558-303-7. Disponível em: <http://doi.acm.org/10.1145/1459359.1459428>.

LIENHARD, J.; LAUER, T. Multi-layer recording as a new concept of combining lecture recording and students' handwritten notes. In: *Proceedings of the 10th ACM International Conference on Multimedia*. New York, NY, USA: ACM, 2002. (MULTIMEDIA '02), p. 335–338. ISBN 1-58113-620-X. Disponível em: <http://doi.acm.org/10.1145/641007.641078>.

LIN, J. et al. A framework-based approach for interactive multimedia application development. In: *Proceedings of the 2012 ACM Research in Applied Computation Symposium*. New York, NY, USA: ACM, 2012. (RACS '12), p. 364–370. ISBN 978-1-4503-1492-3. Disponível em: <http://doi.acm.org/10.1145/2401603.2401683>.

LIU, T.; KENDER, J. Lecture videos for e-learning: current research and challenges. In: *Multimedia Software Engineering, 2004. Proceedings. IEEE Sixth International Symposium on*. [S.l.: s.n.], 2004. p. 574 – 578.

MAUVE, M. et al. Local-lag and timewarp: Providing consistency for replicated continuous applications. *Multimedia, IEEE Transactions on*, 2004. IEEE, v. 6, n. 1, p. 47–57, 2004.

MEIXNER, B.; KOSCH, H. Interactive non-linear video: definition and xml structure. In: *Proceedings of the 2012 ACM Symposium on Document Engineering*. New York, NY, USA: ACM, 2012. (DocEng '12), p. 49–58. ISBN 978-1-4503-1116-8. Disponível em: <http://doi.acm.org/10.1145/2361354.2361367>.

MEIXNER, B.; KöSTLER, J.; KOSCH, H. A mobile player for interactive non-linear video. In: *Proceedings of the 19th ACM international conference on Multimedia*. New York, NY, USA: ACM, 2011. (MM '11), p. 779–780. ISBN 978-1-4503-0616-4. Disponível em: <http://doi.acm.org/10.1145/2072298.2072453>.

MEIXNER, B. et al. Siva suite: authoring system and player for interactive non-linear videos. In: *Proceedings of the international conference on Multimedia*. New York, NY, USA: ACM, 2010. (MM '10), p. 1563–1566. ISBN 978-1-60558-933-6. Disponível em: <http://doi.acm.org/10.1145/1873951.1874287>.

MELO, E. L. et al. WebNCL: a web-based presentation machine for multimedia documents. In: *Proc. Brazilian symposium on Multimedia and the web*. [S.l.]: ACM, 2012. (WebMedia '12), p. 403–410. ISBN 978-1-4503-1706-1.

MüLLER, R.; OTTMANN, T. The "authoring on the fly" system for automated recording and replay of (tele)presentations. *Multimedia Systems*, 2000. Springer-Verlag, v. 8, n. 3, p. 158–176, 2000. ISSN 0942-4962. Disponível em: <http://dx.doi.org/10.1007/s005300000042>.

MORENO, M.; BATISTA, C.; SOARES, L. Ncl and itu-t's standardization effort on multimedia application frameworks for iptv. *ACM, October*, 2010. 2010.

NAGAI, T. Automated lecture recording system with avchd camcorder and microserver. In: *Proceedings of the 37th annual ACM SIGUCCS fall conference*. New York, NY, USA: ACM, 2009. (SIGUCCS '09), p. 47–54. ISBN 978-1-60558-477-5. Disponível em: <http://doi.acm.org/10.1145/1629501.1629512>.

NETO, C. S. S.; PINTO, H. F.; SOARES, L. F. G. Tal processor for hypermedia applications. In: *Proceedings of the 2012 ACM symposium on Document engineering*. New York, NY, USA: ACM, 2012. (DocEng '12), p. 69–78. ISBN 978-1-4503-1116-8. Disponível em: <http://doi.acm.org/10.1145/2361354.2361369>.

NETO, J. R. C. et al. Eventline: representation of the temporal behavior of multimedia applications. In: *Proceedings of the 18th Brazilian symposium on Multimedia and the web*. New York, NY, USA: ACM, 2012. (WebMedia '12), p. 215–222. ISBN 978-1-4503-1706-1. Disponível em: <http://doi.acm.org/10.1145/2382636.2382684>.

NETO, M. C. M.; SANTOS, C. A. An event-based model for interactive live tv shows. In: *Proceedings of the 16th ACM international conference on Multimedia*. New York, NY, USA: ACM, 2008. (MM '08), p. 845–848. ISBN 978-1-60558-303-7. Disponível em: <http://doi.acm.org/10.1145/1459359.1459502>.

OLIVEIRA, A. P.; CAVALCANTE, I. F.; GONCALVES, R. S. O processo de evasÃo (ou desistÊncia) no curso de licenciatura em letras espanhol ofertado pelo campus ead-ifrn: Causas possÍveis. In: *Proc. Simpósio Internacional de Educação a Distância*. [S.l.: s.n.], 2012.

PANG, D. et al. Classx mobile: region-of-interest video streaming to mobile devices with multi-touch interaction. In: *Proceedings of the 19th ACM international conference on Multimedia*. New York, NY, USA: ACM, 2011. (MM '11), p. 787–788. ISBN 978-1-4503-0616-4. Disponível em: <http://doi.acm.org/10.1145/2072298.2072457>.

PEDROSA, D. et al. A multimodal interaction component for digital television. In: *Proceedings of the 2011 ACM Symposium on Applied Computing*. New York, NY, USA: ACM, 2011. (SAC '11), p. 1253–1258. ISBN 978-1-4503-0113-8. Disponível em: <http://doi.acm.org/10.1145/1982185.1982459>.

PIMENTEL, M.; ABOWD, G. D.; ISHIGURO, Y. Linking by interacting: a paradigm for authoring hypertext. In: *Proc. ACM on Hypertext and Hypermedia*. ACM, 2000. (HYPERTEXT '00), p. 39–48. ISBN 1-58113-227-1. Disponível em: <http://doi.acm.org/10.1145/336296.336315>.

PIMENTEL, M. et al. End-user live editing of itv programmes. *International Journal of Advanced Media and Communication*, 2010. Inderscience, v. 4, n. 1, p. 78–103, 2010.

PIMENTEL, M.; JR., L. A. B.; CATTELAN, R. G. Prototyping applications to document human experiences. *IEEE Pervasive Computing*, 2007. IEEE Educational Activities Department, v. 6, n. 2, p. 93–100, abr. 2007. ISSN 1536-1268. Disponível em: <http://dx.doi.org/10.1109/MPRV.2007.40>.

REPPLINGER, M. et al. Extending x3d for distributed multimedia processing and control. In: *Proceedings of the 14th International Conference on 3D Web Technology*. New York, NY, USA: ACM, 2009. (Web3D '09), p. 61–69. ISBN 978-1-60558-432-4. Disponível em: <http://doi.acm.org/10.1145/1559764.1559774>.

RODRIGUES, K. R. d. H. et al. Find: facilitating the identification of intervals and moments for incorporation of additional content in continuous media. In: *Proceedings of the 18th Brazilian symposium on Multimedia and the web*. New York, NY, USA: ACM, 2012. (WebMedia '12), p. 265–268. ISBN 978-1-4503-1706-1. Disponível em: <http://doi.acm.org/10.1145/2382636.2382692>.

ROSS, G. M. What's the use of lectures? - forty years on. *Discourse*, 2011. v. 10, n. 3, p. 23–41, December 2011.

SACERDOTE, H. C. S. AnÁlise do vÍdeo como recurso tecnolÓgico educacional. *REVELLI - Revista de Educação, Linguagem e Literatura da UEG-Inhumas*, 2010. v. 2, n. 1, p. 28–37, 2010. ISSN 1984-6576.

SANTOS, E. M. et al. Evasão na educaÇÃo a distÂncia: Identificando causas e propondo estratÉgias de prevenÇÃo. In: *Proc. Congresso Internacional ABED de Educação a Distância*. [S.l.: s.n.], 2008.

SANTOS, J. A.; MUCHALUAT-SAADE, D. C. Xtemplate 3.0: spatio-temporal semantics and structure reuse for hypermedia compositions. *Multimedia Tools Appl.*, 2012. Kluwer Academic Publishers, Hingham, MA, USA, v. 61, n. 3, p. 645–673, dez. 2012. ISSN 1380-7501. Disponível em: <http://dx.doi.org/10.1007/s11042-011-0732-2>.

SCHNELL MARKUS; SCHMIDT, M. J. M. A. T. G. R. R. V. E. P. B. G. Mpeg-4 enhanced low delay aac - a new standard for high quality communication. In: *Audio Engineering Society Convention 125*. [s.n.], 2008. Disponível em: <http://www.aes.org/e-lib/browse.cfm?elib=14656>.

SCHULTE, O. A.; WUNDEN, T.; BRUNNER, A. Replay: an integrated and open solution to produce, handle, and distributeaudio-visual (lecture) recordings. In: *Proceedings of the 36th annual ACM SIGUCCS fall conference: moving mountains, blazing trails*. New York, NY, USA: ACM, 2008. (SIGUCCS '08), p. 195–198. ISBN 978-1-60558-074-6. Disponível em: <http://doi.acm.org/10.1145/1449956.1450016>.

SCHWABER, K.; BEEDLE, M. *Agile software development with Scrum*. [S.l.]: Prentice Hall, 2001.

SCHWERDT, G.; WUPPERMANN, A. C. Sage on the stage: Is lecturing really all that bad? *Education Next*, 2011. v. 11, n. 3, p. 62–67, 2011.

SOARES, L. *Programando em NCL 3.0: desenvolvimento de aplicações para middleware Ginga: TV digital e Web*. [S.l.]: Elsevier, 2009.

SOARES, L.; MORENO, M.; NETO, C. D. S. S. Ginga-ncl: Declarative middleware for multimedia iptv services. *Communications Magazine, IEEE*, 2010. IEEE, v. 48, n. 6, p. 74–81, 2010.

SOARES, L.; RODRIGUES, R. Nested context language 3.0 part 8–ncl digital tv profiles. *Monografias em Ciência da Computação do Departamento de Informática da PUC-Rio*, 2006. v. 1200, n. 35, p. 06, 2006.

SOARES, L.; RODRIGUES, R.; MORENO, M. Ginga-ncl: the declarative environment of the brazilian digital tv system. *Journal of the Brazilian Computer Society*, 2007. SciELO Brasil, v. 12, n. 4, p. 37–46, 2007.

SOARES, L. F. et al. Variable and state handling in ncl. *Multimedia Tools Appl.*, 2010. Kluwer Academic Publishers, Hingham, MA, USA, v. 50, n. 3, p. 465–489, dez. 2010. ISSN 1380-7501. Disponível em: <http://dx.doi.org/10.1007/s11042-010-0478-2>.

TEIXEIRA, C. et al. Taking advantage of contextualized interactions while users watch tv. *Multimedia Tools and Applications*, 2010. Springer, v. 50, n. 3, p. 587–607, 2010.

TONNDORF, K. et al. Challenges in creating multimedia instructions for support systems and dynamic problem-solving. In: *Proceedings of the 12th International Conference on Knowledge Management and Knowledge Technologies*. New York, NY, USA: ACM, 2012. (i-KNOW '12), p. 33:1–33:4. ISBN 978-1-4503-1242-4. Disponível em: <http://doi.acm.org/10.1145/2362456.2362497>.

VAISHNAVI, I. et al. From iptv to synchronous shared experiences challenges in design: Distributed media synchronization. *Sig. Proc.: Image Comm.*, 2011. p. 370–377, 2011.

VALIN, J.-M.; TERRIBERRY, T. B.; MAXWELL, G. A full-bandwidth audio codec with low complexity and very low delay. In: *17th European Signal Processing Conference. Glasgow, Scotland*. [S.l.: s.n.], 2009.

VEGA-OLIVEROS, D. A.; MARTINS, D. S.; PIMENTEL, M. d. G. C. "this conversation will be recorded": automatically generating interactive documents from captured media. In: *Proceedings of the 10th ACM symposium on Document engineering*. New York, NY, USA: ACM, 2010. (DocEng '10), p. 37–40. ISBN 978-1-4503-0231-9. Disponível em: <http://doi.acm.org/10.1145/1860559.1860568>.

VIEL, C. C. et al. Multimedia presentation integrating interactive media produced in real time with high performance processing. In: *Proceedings of the 18th Brazilian symposium on Multimedia and the web*. New York, NY, USA: ACM, 2012. (WebMedia '12), p. 115–122. ISBN 978-1-4503-1706-1. Disponível em: <http://doi.acm.org/10.1145/2382636.2382664>.

VIEL, C. C. et al. Go beyond boundaries of dtv applications. In: *Proceedings of the 2013 ACM Symposium on Document Engineering*. New York, NY, USA: ACM, 2013. (DocEng '13).

VIEL, C. C. et al. How are they watching me: learning from student interactions with multimedia objects captured from classroom presentations. In: *Proc. International Conference on Enterprise Information Systems*. [S.l.: s.n.], 2013. (ICEIS '13).

VIEL, C. C. et al. Presentations preserved as interactive multi-video objects. In: *Proc. Workshop on Analytics on Video-Based Learning*. [S.l.: s.n.], 2013.

VIEL, C. C. et al. Rv-mtv: Framework para interação multimodal com aplicações de realidade virtual em tv digital e dispositivos móveis. In: *WebMedia 2011*. [S.l.: s.n.], 2011.

VIEL, C. C. et al. An approach for controlling synchronous remote instances of a multimedia presentation. In: *Submmited to a conference*. [S.l.: s.n.], 2013.

VUORIMAA, P. *Timesheets JavaScript Engine*. 2007.

VUORIMAA, P.; BULTERMAN, D.; CESAR, P. Smil timesheets 1.0. *W3C Working Draft*, 2008. 2008.

WEISZ, J. D. et al. Watching together: integrating text chat with video. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. New York, NY, USA: ACM, 2007. (CHI '07), p. 877–886. ISBN 978-1-59593-593-9. Disponível em: <http://doi.acm.org/10.1145/1240624.1240756>.

GLOSSÁRIO

AAC-ELD – *Mpeg-4 Advanced Audio Coding Enhanced Low-Delay*

API – *Application Programming Interface*

AVCHD – *Advanced Video Coding High Definition*

AVC – *Mpeg-4 Advanced Video Coding*

BML – *Broadcast Markup Language*

CSS – *Cascading Style Sheets*

DOM – *Document Object Model*

DSM-CC – *Digital Storage Media — Command and Control*

DTV – *TV Digital*

EE – *Electronic Learning*

EPG – *Electronic Program Guides*

EaD – *Educação a Distância*

HTML – *HyperText Markup Language*

HTTP – *Hypertext Transfer Protocol*

IPTV – *Internet Protocol TV*

ISDB-T – *Integrated Services Digital Broadcasting*

LMS – *Learning Management System*

MMVC – *Multimedia-View-Controller*

MOOC – *Massive Open Online Course*

MVC – *Model-View-Controller*

NCL – *Nested Context Language*

RTMP – *Real-Time Messaging Protocol*

RTPC – *Real-Time Control Protocol*

RTP – *Real-Time Transport Protocol*

RTSP – *Real-Time Streaming Protocol*

SMIL – *Synchronized Multimedia Integration Language*

STB – *Set-Top Box*

TAL – *Template Authoring Language*

TIC – *Tecnologias da Informação e Comunicação*

VOIP – *Voice Over Internet Protocol*

VR – *Virtual Reality*

XHTML – *Extensible HyperText Markup Language*

XML – *Extensible Markup Language*

iDTV – *Interactive Digital TV*

iDTV – *Interactive Digital TV*